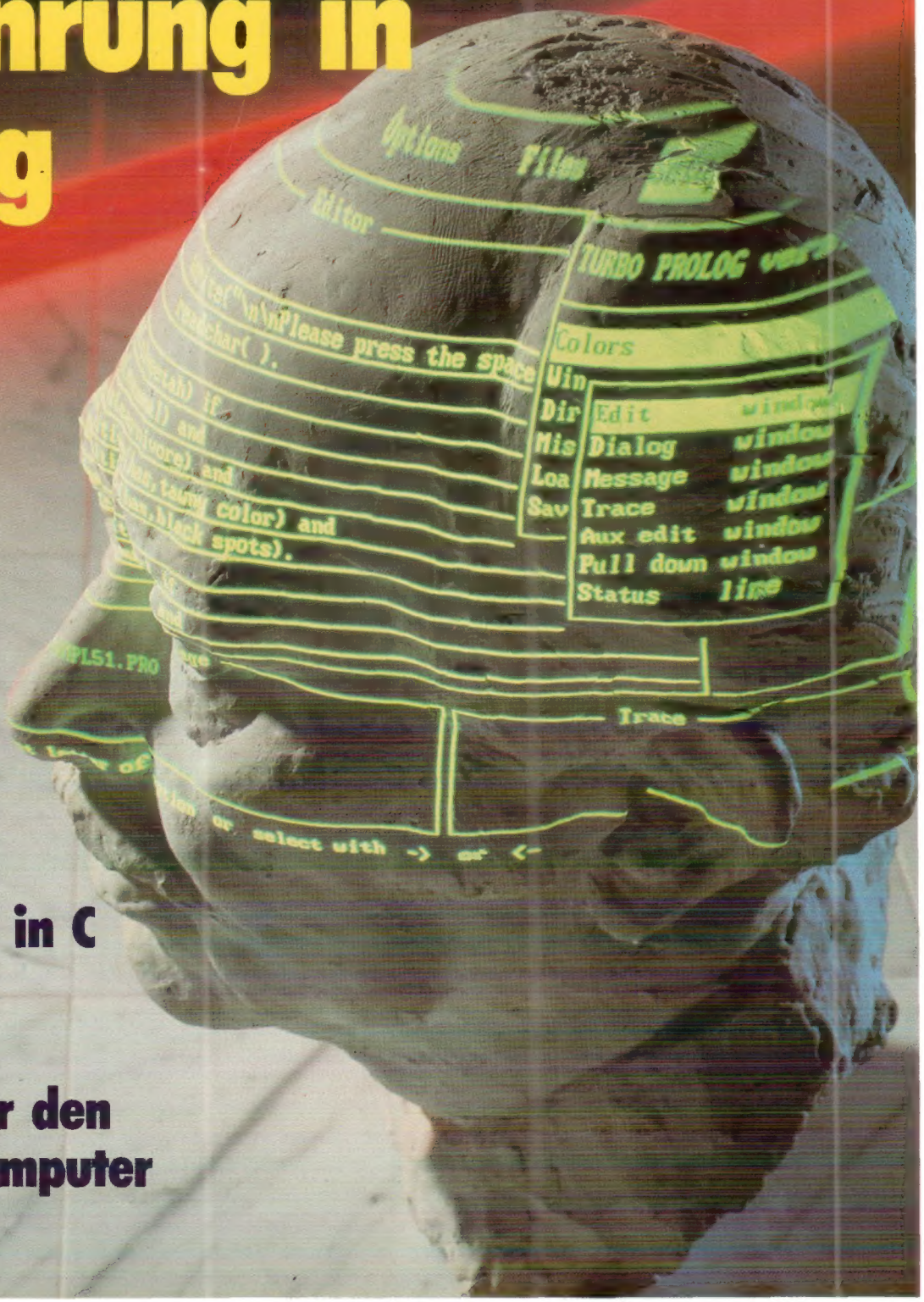
mc
5
Die Mikrocomputer-Zeitschrift
7 DM · 60 öS · 7 sfr. · Mai 1987

Software-Technik:
Turbo-Pascal transparent

Künstliche Intelligenz:
Einführung in Prolog



Grundlagen:
Gleitkomma-zahlen in Pascal

Atari ST:
RAM-Monitor

Software:
Unix-Move für MS-DOS
Grafik-Paket in C

Selbstbau:
Z80-Karte für den mc-68000-Computer

Liebe Leser! Fleißige Biber!

Gleich nach Erscheinen unserer Ausgabe 4, zu einem Zeitpunkt, zu dem nur wirklich passionierte Leser auch schon in mc hineingeschaut haben konnten, klingelte bei mir das Telefon. Wie das denn mit den fleißigen Bibern sei, wurde ich befragt, es sei doch ganz einfach, mit einer Schleife unendlich viele Einsen zu erzeugen. Natürlich war es so nicht gemeint – und das stand auch ganz klar in der Ausschreibung: Nur Maschinen, die wirklich auch einmal das Rechnen von selbst abbrechen, sind gültig. Aber dann: Wie das mit den Einsen gemeint sei, ob die denn irgendwie verteilt sein dürften oder ohne Nullen dazwischen ununterbrochen aufeinander folgen sollten?

Das war in der Tat eine Unklarheit in der Ausschreibung, die besonders für Leute, die noch nie einen fleißigen Biber gesehen haben, schlimme Folgen haben könnte. Also sei hier ganz klar ausgesprochen, daß die Folge der produzierten Einsen nicht durch Nullen unterbrochen sein darf. Weiter wurde gefragt, wie denn der Anfangszustand des Bandes sein müsse. Die Antwort: es soll leer sein, also nur Nullen enthalten. Und weiter wurde noch nachgefragt, ob es denn irgendwie rechts oder links begrenzt sei. Nochmals die ganz klare Aussage: Das Rechenband der Maschine soll sowohl nach rechts als auch nach links nicht begrenzt sein.

Hoffentlich haben diese Präzisierungen jetzt alle Zweifel beseitigt. Wer nun gleich fleißig in die vielleicht falsche Richtung losgerechnet hat – oder losrechnen hat lassen, der sollte nicht enttäuscht sein. Die fleißigen Biber mit den fünf Zuständen sind ein hinreichend kompliziertes Problem, für das Arbeiten auch „in die falsche Richtung" nicht umsonst aufgewendet sind.

Es gibt einen Amiga zu gewinnen und viele Franzis-Bücher nach eigener Wahl. Wir glauben also, daß genügend materieller Anreiz vorhanden ist, die Biber fleißig werden zu lassen. Abgesehen davon ist das Problem auch rein intellektuell, ohne Preis, eine Herausforderung. In den USA ist die Suche nach fleißigen Bibern weit fortgeschritten, nach unseren Informationen gibt es einen mit 5 Zuständen, der 1915 Einsen produziert. Das muß ich hier öffentlich bekannt geben, weil ich das auch schon jemandem am Telefon mitgeteilt habe. Die Chancengleichheit muß ja gewahrt bleiben.

Was geschieht nun mit jemandem, der genau diesen Biber in der Literatur auftut und diesen abgekupferten uns schickt? Es ist wie im richtigen Leben, wer zur rechten Zeit weiß, wo er nachschauen muß, der gewinnt eben auch. Aber sportlicher wäre natürlich ein Biber aus Eigenmitteln. Und stellen Sie sich den Ruhm vor: Wir nennen ja den Namen des Biberzüchters öffentlich. Zu diesem Problemkreis noch weiter: Die Biberitis hört mit dem gefundenen rechenkräftigsten Biber nicht auf. Noch ist unseres Wissens nicht klar, ob es überhaupt einen Biber mit fünf Zuständen gibt, der eine maximale Anzahl von Einsen produziert. Wenn es ihn gibt, bleibt die Frage, ob er der Einzige ist. Und wenn das alles klar ist, dann kann man natürlich nach den Bibern mit sechs vom Stop-Zustand verschiedenen Zuständen fragen. Bis dahin ist es aber noch weit, zunächst viel Spaß mit den Bibern mit fünf Zuständen.

Ihr

Ulrich Rohde

mc-inhalt

mc-editorial

Liebe Leser
Fleißige Biber! 3

mc-briefe 6

mc-info 8

Wettbewerb für Computerkünste in Linz 116

Spruch des Monats 100

Zitat des Monats 63

mc-bücher 42

Neues vom Buchmarkt 91

mc-grundlagen

Prolog – eine Einführung 54
Bevor man mit dem Programmieren in Logik beginnen kann, muß man sich mit den Prinzipien auseinandersetzen, nach denen komplexe Probleme gelöst werden. Wir bringen eine Einführung und zeigen, wie ein Differentiationsprogramm in Prolog aussieht

Die gläserne Gleitkommazahl 60
Wir stellen ein Turbo-Pascal-Programm zur Untersuchung der internen Gleitkommazahlendarstellung vor

Datenbanksysteme – Teil 1 64
In einer umfangreichen Artikelserie bringen wir alles Wissenswerte über Datenbanksysteme. In dieser Ausgabe erklären wir die wichtigsten Begriffe und zeigen Vor- und Nachteile bestimmter Datenbankmodelle auf

Hinter den Kulissen von Pascal 68
Was passiert eigentlich im Computer, wenn er ein Pascal-Programm compiliert und ausführt? Dieser Frage gehen wir nach und zeigen, wie Pascal-Strukturen in Maschinensprache übersetzt werden

mc-soft

Quadratwurzel-Algorithmus 58
Eine Routine in Turbo-Pascal ist 30mal schneller als die Standardfunktion SQRT

Druckerumschaltung durch Type-Befehl 72
Außer über Schalter am oder sogar im Drucker kann man die Schriftarten auch durch Software auswählen

Cross-Assembler und EPROM 74
Ein Programm für alle, die Dateien von Cross-Assemblern in EPROMs brennen wollen

MS-DOS-Funktionstasten umbelegen 76
Für alle Siemens-PC-D-Anwender

Funktionsinterpreter in Turbo-Pascal 78
Beliebige Zeichenketten werden analysiert und übersichtlich aufgeschlüsselt dargestellt

Move-Utility für MS-DOS 82
Mit dieser nützlichen Assembler-Routine stellen wir eine weitere Anwendung des Interrupt 21H vor

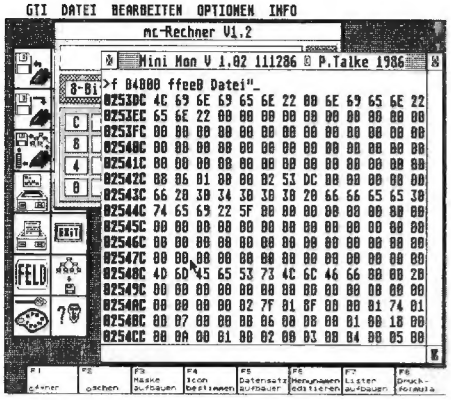
Titelfoto: Klaus Hager Büste: Franz Leschinger
(Staatliche Akademie der Bildenden Künste, München)

Prolog

Die Programmiersprache Prolog ist anders als andere Programmiersprachen. Damit Ihnen der Einstieg in das Programmieren mit Logik leichter fällt, bringen wir eine Einführung mit vielen Beispielen. **Seite 54**

PS/2 von IBM

Mit fast 100 neuen Produkten, die unter der Bezeichnung Personal System /2 vorgestellt wurden, will IBM zeigen, wie es nach der PC-Ära weitergeht. Ein erster Bericht, auch mit Informationen über das neue Multitasking-Betriebssystem, ab **Seite 12**



RAM-Editor für Atari ST

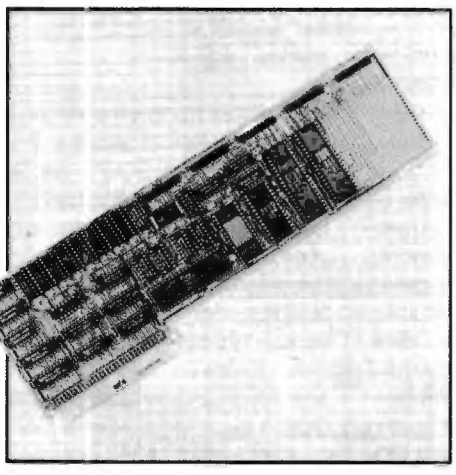
Wir bringen ein nützliches Accessory für den Atari ST, damit sich jeder Anwender über die Belegung des Speichers informieren kann und an Bereiche seines Rechners kommt, die unter GEM-Software ansonsten verborgen bleiben. Dieses Programm hilft beim Debuggen, beim Installieren von Utilitys und vielem mehr.

Seite 102

Z80-Karte für den mc-68000-Computer

Für den mc-68000-Computer bringen wir eine Steckkarte mit einer Z80-CPU, die externe Prozesse kontrolliert und Peripheriegeräte steuert. Wie immer in mc sehr ausführlich und verständlich beschrieben.

Seite 119



Zeitanzeige in Fenstertechnik 86
In einem Window mit frei wählbarer Position zeigen alle PCs mit einem Assembler-Programm und einer Echtzeituhr die genaue Zeit an

Das andere Chaos 88
Für alle GFA-Basic-Anwender ein kleines Programm, das auf dem Atari ST sehr schnell hübsche Grafiken erzeugt

Apple-CP/M mit der Maus „booten“ 90
Nach einem kleinen Patch bootet der Apple-II auch dann, wenn eine Mauskarte eingesteckt wurde

Grafik-Paket in C 92
Wer immer wieder mathematische Funktionen oder Meßwerte grafisch darstellen muß, wird sich über dieses universelle C-Programm freuen. Ein Beitrag mit vielen Beispielen

RAM-Monitor unter GEM für den Atari ST 102
Als Accessory steht unser Minimonitor, der das Ansehen und Ändern, Laden und Speichern von Daten erlaubt, unter allen GEM-Programmen stets zur Verfügung

Die CD-ROMs kommen 105

GW-Basic mit DOS-Komfort 106
Durch den Shell-Befehl kann man von Basic aus alle MS-DOS-Kommandos ausführen lassen. Neben der Erstellung sortierter Directories wandelt das Programm Wordstar-Dateien in ASCII-Dateien um

Software für Videodat-300 109
Der Schneider CPC-664 empfängt Daten, die mit dem Fernsehbild übertragen werden, speichert sie auf Diskette ab und zeigt sie am Bildschirm an

Abfangen von Laufwerksfehlern unter CP/M 111
Diesen Artikel sollten alle lesen, die nicht möchten, daß ihre Programme bei fehlerhaften Diskettenzugriffen abgebrochen werden

Preiswerte Software für den PC 112

PC-Grafik-Symbole 116

Der mc-68000-Computer lernt griechisch 118

mc-test

SAL: Ein Software-Entwicklungswerkzeug 113
Auf dieses Programm haben viele Programmierer gewartet. Es vereinfacht die Software-Entwicklung mit zahlreichen Compilern und stellt nützliche und immer wiederkehrende Funktionen zur Verfügung

mc-hard

LAN und Telekommunikation 83

Programmierbare Logikbausteine 89

VIA 6522 am NDR-68008 117
Anwender eines 68008-Systems können die preiswerte VIA 6522 mit einer Synchronisierschaltung als Schnittstelle zur Umwelt einsetzen

Z80-Karte für den mc-68000-Computer 119
Eine Steckkarte für den mc-68000-Computer, mit der externe Prozesse kontrolliert und Peripheriegeräte gesteuert werden

mc-markt 128

mc-vorschau 166

Impressum 164

Teurer Flug über den Teich?

Als aufmerksamer Leser Ihrer Zeitschrift sowie amerikanischer Computerzeitschriften muß ich immer wieder feststellen, für wie dumm uns doch deutsche Händler halten müssen. Da werden USA-Programme für 500 $ Enduser-Preis angeboten und hier in Deutschland soll man dann plötzlich 2000 DM für das Programm bezahlen. Und das bei einem Dollarkurs von unter 2 DM!
Freilich ist die Fracht zu berechnen; beinahe zu vernachlässigen ist hingegen der Zoll von 2.50 DM pro bespielter Diskette. Über den Support kann man sich streiten – der wird auch in den USA geleistet. Sind vielleicht die hohen Anzeigenpreise bei uns dafür verantwortlich, daß wir hier das Doppelte und bei manchen Programmen noch mehr bezahlen müssen? Einer muß ja schließlich Schuld sein und entsprechende Ausreden kann man sich in den verschiedensten Variationen anhören! Mich würde interessieren, wie andere Leser darüber denken. *Hans Luszeit*
1000 Berlin 42

Kommentar der Redaktion: Hohe Preise sind wirklich ein Ärgernis. Aber Sie dürfen nicht alle Fälle über einen Kamm scheren. Bedenken Sie zum Beispiel, daß man Lizenzen mit US-Firmen meist nur bekommt, wenn man deren Bosse persönlich besucht und nur behält, wenn man regelmäßigen Kontakt hält. Das kostet viel Geld.

Fleißige Biber, erste Frage

Aus ihrer Aufgabenstellung zum Wettbewerb „Fleißige Biber" in mc 4/87 geht nicht klar hervor, welche Anzahl von Einsen Sie nun wissen wollen: Geht es um die Einsen, die der Biber auf das Rechenband schreibt oder um jene, die beim Abbruch auf dem Rechenband stehen? Sind im Beispiel 2 die fünf, die auf das Band geschrieben wurden, gemeint oder aber die vier auf dem Band stehenden?
Dieter Neumann
4785 Borchen

Fleißige Biber, zweite Frage

Ich möchte an Ihrem Wettbewerb teilnehmen, habe aber das Problem der fleißigen Biber nicht ganz verstanden. Daher suchte ich in früheren Ausgaben von mc Rat. Das Prinzip einer Turing-Maschine habe ich in mc 2/86 nachgelesen und wie ich glaube, auch verstanden. Ein Problem aber bleibt: Was verstehen Sie unter einem Rechenband voller Einsen? Wann wird eine Eins geschrieben und wann nicht?
Michael Buss
4530 Ibbenbüren

Fleißige Biber, Antworten

Zunächst kommt es auf die nach dem Stoppen auf dem Band stehenden Einsen an, nicht auf die, die möglicherweise unterwegs beim Rechnen entstehen und vergehen. Eine Eins wird von der Turingmaschine immer dann geschrieben, wenn sie so konstruiert ist, daß sie bei erfüllter Eingangsbedingung eben eine Eins schreibt, wenn also der „dann-Teil" der Tafel, den Sie als Programmierer vorgeben, dies vorschreibt. Das Ziel des fleißigen Bibers sollte es sein, das Rechenband mit einer möglichst großen Anzahl von Einsen zu füllen, die noch dazu nicht voneinander durch Nullen getrennt sein dürfen.

Fehler im Patchprogramm

Beim Abdruck der von Reinhard Fischer in mc 11/86 vorgeschlagenen Änderung des Appleworks-Patchprogramms ist anscheinend ein Mißgeschick unterlaufen! Der dort vorgeschlagene Patch überschreibt einen anderen Programmteil an der Adresse $57A, so daß hinterher gar nichts mehr funktioniert. Außerdem wurde im Listing gegenüber der Originalzeile das erste Byte des Strings meiner Ansicht nach grundlos von $7F auf $CC geändert. Daher habe ich in meinem Patchprogramm die Zeile 310 folgendermaßen modifiziert.

```
310 DATA 2C FF CF 2C
    58 FF 3E 85 40 0A 0A 18 65
    40 85 40 A5 41 0A 26 41 0A 26
    41 18 65 3E 85 16 4C C7 20
    ' GOSUB 10: REM Patch
```

nach wie vor aus mc 11/86 (korrigiert).

Dieser funktioniert auf meinem Computer wieder alles. Ob der Apple-Controller auch mitspielt, kann ich nicht prüfen. Wenn aber die Idee des Patches von Reinhard Fischer stimmt, müßte es nun klappen.
Marcus Gröber
5090 Leverkusen 1

Unbegründete Kritik

In mc-Ausgabe 2/86, kam Dr. Peter Dörre zu Wort, der seine negativen Erfahrungen mit dem Atari ST schilderte. Wie ich feststellen konnte, schaffte es Dr. Dörre, daß auch zwei andere Computer-Zeitschriften denselben Artikel abdruckten. Schon allein deshalb, weil der Verfasser gleich drei Zeitschriften vorwirft, einen Computer falsch zu beurteilen, erscheinen seine Aussagen unglaubwürdig.
Zu Dr. Dörres Kritikpunkten Stellung nehmend, sei festgehalten:
Jeder Computer kann defekt sein. Der schlechte Service, der Dr. Dörre zuteil wurde, ist dem Fachhändler vorzuwerfen, nicht Atari. Das genannte „Softwareproblem" existiert nicht. Einschlägige Software-Übersichten beweisen dies. Außerdem sind für den ST neue Basic- und Logo-Versionen Public Domain. Für den ST gibt es ein Unzahl von PD-Programmen, u.a. Forth, Lisp, Prolog, Diskettenmonitore, Grafikprogramme, Textverarbeitung, Datenverwaltungsprogramme und Utilities, um nur einige zu nennen. Allein die große Zahl von kostenlosen Desk-Accessories wertet das Atari-GEM erheblich auf – was in keinem Test berücksichtigt wurde.
Patrick Schmitz
5480 Remagen-Unkelbach

Modifikationen machen's möglich!

Die in mc 2/87 vorgestellte Turbo-Taktumschaltung (S. 61) ist auf meinem Turbo-Board (No Name Taiwan) nicht lauffähig. Bei meinen System darf nur das Bit 2 (D2) gesetzt sein. Im folgenden sind alle Änderungen durch <---- markiert:

```
•
•
in al,61h
xor al,4 <----
•
•
mov cx,0607h
and al,4 <----
```

Die in mc 2/87, S. 50 beschriebene DOS-EXEC Funktion von Turbo-Pascal aus läuft mit einigen Änderungen bei mir einwandfrei: In der Procedure exec nach inline

```
with reg do
begin
•
•
bx:=ofs (Par–block);
ax:=$4a00; <----
msdos (reg); <----
```

Schließlich noch eine Anmerkung zum Artikel „Cursor ändern unter MS-DOS", der in mc 3/87, S. 86 erschienen ist. Um den Cursor wieder auf „normale" Größe zu bringen, bedarf es bei meinem Computer folgender Schritte:

```
cursor–normal: mov ch,6 <---
mov cl,7 <----
```

Alle Änderungen laufen unter DOS 3.2. *Elmar Ohst*
4030 Ratingen

PC für weniger als 2000 DM von Sanyo, der MBC 16 Plus

Sanyo stellt Low-Cost-PC vor

Zur CeBIT konnte Sanyo ein neues Mitglied der PC-Elite-Familie vorstellen, den MBC 16 Plus. Der Rechner ist mit einer 8088-CPU ausgestattet, deren Taktfrequenz von 4,77 MHz auf 8 MHz umgeschaltet werden kann. In der Ausführung mit einem Floppylaufwerk und einem Monitor kostet der MBC 16 Plus 1935 DM, ohne Monitor ist man ab 1666 DM dabei.

Neben 640 KByte RAM bietet der MBC 16 Plus acht PC-Steckplätze, eine parallele und eine serielle Schnittstelle sowie eine Videokarte, die Hercules- (720 x 350 Punkte) und CGA-(640 x 200 Punkte) Kompatibel ist.

Die Rechner werden mit DOS 3.2 und GW-Basic geliefert, eine 20-MByte-Festplatte ist optional.

Tragbarer mit Festplatte

Mit gut 9 kg wiegt der neue Compaq-Portable-III zwar fast doppelt so viel wie viele seiner Kollegen, dafür steht er aber auch ausgewachsenen Tischcomputern in nichts nach. Seine CPU, ein 80286, läuft mit 12 MHz und ist damit schneller als die des Standard-AT; bei Bedarf läßt sie sich auch auf 8 MHz drosseln. Das

Schwerer, aber auch leistungsfähiger ist der neue Portable-III im Vergleich zu vielen Konkurrenzprodukten; das Gerät ist ab etwa 12500 DM zu haben

relativ hohe Gewicht kommt
stande, weil Compaq nicht
andere Firmen auf einen
Massenspeicher verzichtet h
genteil: Das 5,25-Zoll-Disket
wird ergänzt von einer Harddi
weder 20 MByte (Modelle 2
MByte (Modell 40) speichern
Laufwerke haben eine mittlere
von 30 ms und sind stoßsicher
Bei der Diskettenkapazität h
Wahl zwischen 1,2 MByte und
Der Portable-III mißt 406 mm
198 mm (B x H x T). Im Grund
ken 640 KByte RAM, die ohne
tinen auf 6,6 MByte erweitert w
nen. Auch die nötigsten So
(seriell, parallel, RGBI) sind
handen. Ein Modem läßt sich
für die Bundesrepublik Deut
diese Option aus postalische
allerdings noch nicht erhältlic
Unterschiedlich gegenüber de
tragbaren Computern ist auch
geeinheit: Statt der üblicher
stallanzeige hat Compaq ein
Plasma-Bildschirm spendiert
Grafikmodus 640 x 400 Punk
Betriebsart, die kompatibel zu
dard-Grafik ist, gibt es eben
schattierte Darstellungen si
Nach Angaben des Herstelle
zum erstenmal bei einem
schirm verwirklicht.
Wer im neuen Computer Zusa
treiben will, kann dies mit H
weiterungsbox tun, die auf d
rät huckepack aufgesteckt w
zwei AT-kompatible Einste
Verfügung.

Apple erfolgreich

Apple Computer erreichte in
tal des Geschäftsjahres 198
plus von 24 Prozent. Die Ge
um knapp zwei Millionen Do
samt 58,5 Millionen Dollar
für die positive Entwicklu
Akzeptanz des Apple-IIGS
kordverkauf des Macintos
Den Anreiz am Apple-IIGS
nehmen mit einer Aktion z
gen Bestehen weiter steig
schen Versionen der Origi
heiten des Apple-IIe, Ap
und ITT 2020 können für
von 1795 DM gegen einen
256 KByte RAM und Maus
den. Für Schulen werden
mäßigungen angeboten.

**Teur
über**

Als aufm
Zeitschrif
scher
muß ich
stellen,
doch deu
müssen.
gramme
Preis an
Deutschl
plötzlich
gramm b
einem D
DM!
Freilich i
rechnen;
lässigen
von 2.5(
Diskette.
kann ma
wird auc
stet. Sin
Anzeiger
verantwo
das Dop
chen Pro
bezahler
ja schlie
entsprec
kann m
schieder
hören! M
ren, wie
denken.

*Kommer
Hohe Pr
Ärgernis
alle Fäll
scheren.
Beispiel,
mit US-
kommt,
se persö
behält,
gen Kor
viel Gelc*

**Fleiß
erst**

Aus ih
zum We
ber" in n
hervor,
Einsen
Geht es
Biber a
schreibt
beim Al

Neue PC-Familie von IBM

Fünf Jahre nach der Vorstellung des IBM-PC und zeitgleich in den USA und Europa stellte die IBM ihre neue PC-Familie Personal System /2 vor, die vier Computermodelle und insgesamt etwa 100 Produkte umfaßt. Mit den Computern, für die IBM alleine in den USA Werbung für 50 Mill. $ machen wird, soll das Unternehmen aus den zuletzt rückläufigen Verkaufszahlen herauskommen. Am unteren Ende der Preis- und Leistungsklasse findet sich das Modell 30, das auf einer 8086-CPU basiert und etwa 3545 DM kosten soll. Am oberen Ende ist das Modell 80 angesiedelt, das mit einem 80386 arbeitet und ab 15162 DM zu haben sein soll. Dazwischen liegen die Modelle 50 und 60, die mit der CPU 80286 arbeiten. Alle neuen Computer sind in Surface Mounted Technologie (SMT) aufgebaut. Dadurch wurde erreicht, daß die Standard-Schnittstellen auf der Systemplatine aufgebaut werden konnten. Als Speicher-ICs kommen 1-MBit-Chips aus dem Sindelfinger Werk zum Einsatz.

Multitasking-Betriebssystem

Als Betriebssystem bietet IBM für die neuen Modelle das DOS 3.3 an, das in einigen Punkten gegenüber dem DOS 3.2 verbessert wurde. Ab dem ersten Quartal 1988 soll dann ein Multitasking-Betriebssystem kommen, das auf dem Betriebssystem 2 von Microsoft aufbaut. Die Betriebssystemsoftware /2 wird voraussichtlich etwa 1800 DM kosten. Neben der Standardversion, die eine grafische Bedienerführung enthält, wird es eine erweiterte Version mit Kommunikations- und einem relationalen Datenbank-Manager geben. Das Betriebssystem /2 kann einen Hauptspeicher von maximal 16 MByte direkt adressieren und Dateien von mehr als 32 MByte auf Festplatten ansprechen. Das Betriebssystem /2 umfasst etwa 5 MByte und kann auf den neuen Modellen 50, 60 und 80 sowie auf dem IBM AT und dem IBM XT286 eingesetzt werden.

Einstieg bei 4000 DM

Das Modell 30 wird in zwei Versionen angeboten; einmal mit zwei 3½-Zoll-Floppylaufwerken (3545 DM) und einmal mit einem 3½-Zoll-Floppylaufwerk und einer 20 MByte Festplatte (5000 DM). Die Kapazität einer Diskette beträgt 720 KByte. Die CPU 8086 wird mit 8 MHz ohne Wartezyklen getaktet. Der Hauptspeicher ist mit 640 KByte ausgerüstet und kann bis auf 2,64 MByte erweitert werden. An Anschlüssen finden sich bei diesem Modell eine serielle (75...19200 Baud), eine bidirektionale parallele (19200 Baud) und eine Maus-Schnittstelle. Über drei 8-Bit-Steckplätze kann man dieses Modell weiter ausbauen. Der Ärger mit dem Einstellen zahlreicher Dip-Schalter gehört der Vergangenheit an, da alle Funktionskarten elektronisch erkannt werden. Neben einer batteriegepufferten Uhr hat das Modell 30 eine Fassung für den mathematischen Coprozessor 8087. An Bildschirmen bieten sich ein monochromer und drei farbige Typen zur Auswahl. Alle vier Bildschirme sind völlig neu entwickelte analoge Typen und zeichnen sich durch hervorragende Qualitätsmerkmale aus. Sie werden am MCGA-Anschluß (Multi Color Graphics Array) des Computers betrieben, der zum CGA kompatibel ist, und eine maximale Auflösung von 640 x 480 Punkten erreicht. Auf dem monochromen schwarz/weiß 12-Zoll-Monitor mit einer Bildwiederholfrequenz von 70 Hz und 88 Pixeln je Zoll sind 64 Grautöne darstellbar. Das Modell 30 kann nicht mit dem Betriebssystem /2 arbeiten. Bei diesem Computer kommt DOS 3.3 zum Einsatz, das zu DOS 3.2 kompatibel ist. Der Interrupt 21H bekommt in dieser Version neue Funktionen, die zum Beispiel erlauben, mehr als 20 Dateien gleichzeitig zu öffnen. Recht interessant auch der residente Befehl Fastopen, der Dateien, die kurz zuvor geöffnet waren, sehr schnell wieder öffnen kann. Dadurch sollen Leistungssteigerungen von bis zu 15% erzielt werden. Zum DOS 3.3 gehört auch ein neues Basic. Die Programme Linker, Exe2bin und Vdisk, die bislang mit dem DOS geliefert wurden, gehören jetzt zum Lieferumfang des technischen Handbuches.

Wer von einem PC auf das Modell 30 oder eines der anderen Modelle umsteigen möchte, kann seine Daten über die serielle, aber auch über die parallele Schnittstelle übertragen. Disketten im 5¼-Zoll-Format sind über ein externes 5¼-Zoll-Laufwerk weiter verwendbar.

Im Gegensatz zum Modell 30, das sofort verfügbar sein soll, wird es das Modell 50 erst ab Ende Mai geben. Das Modell 50 ist das Basismodell der System /2-Familie mit einer neuen Systemarchitektur, Mikro-Kanal genannt. In diesem Computer, der ab 8500 DM zu haben sein wird, kommt ein 80286-Prozessor mit 10 MHz Taktfrequenz zum Einsatz. Der Hauptspeicher ist mit 1 MByte RAM (150 ns) ausgestattet, eine Erweiterung bis auf 7 MByte ist intern möglich. Dieses Modell ist mit einer 20-MByte-Festplatte und einem 3½-Zoll-Laufwerk ausgestattet, das eine Kapazität von 1,44 MByte erreicht. Die in diesem Computer zum Einsatz kommende Mikro-Kanal-Architektur sorgt für eine optimale Auslastung des Prozessors und besteht hauptsächlich aus einem 16-Bit-DMA-Datenbus, der 8 MByte in der Sekunde übertragen kann. An diesen Kanal sind alle Peripheriegeräte angeschlossen, die somit nicht mehr vom Prozessor direkt gesteuert werden müssen. Das Modell 50 kann über drei 16-Bit-Steckplätze ausgebaut werden. An diesen Computer können die gleichen Monitore wie an das Modell 30 angeschlossen werden, nur erreichen sie beim Modell 50 bessere Leistungsdaten. Dafür ist der Monitoran-

Das Einstiegsmodell 30 bietet modernste Technik auf kleinstem Raum

Mit dem Modell 80 will IBM im nächsten Jahr hauptsächlich in die Bereiche Forschung und Entwicklung vordringen

schluß VGA (Video Graphics Array) verantwortlich, der CGA- und EGA-Kompatibel ist. Maximal sind mit diesem neuen Standard 262144 Farben und eine maximale Auflösung von 720 x 400 Punkten erreichbar. Mit einem zusätzlichen Adapter und dem 16-Zoll-Farbmonitor können 256 Farben bei einer Auflösung von 1024 x 768 Punkten erreicht werden. Weitere Adapter für dieses Modell dienen der Kommunikation mit IBM-Großrechnern, die nun ohne besonderen Aufwand direkt mit dem PC-ähnlichen Personal System /2 zusammenarbeiten können. Auf dem Modell 50 kann neben dem DOS 3.3 das neue Betriebssystem /2 zur Anwendung kommen. Auch beim BIOS hat IBM einige Verbesserungen vorgenommen. So erkennt der Rechner beim Booten defekte Speicheradressen und umgeht sie beim späteren Betrieb. Noch mehr Speicherplatz (max. 15 MByte RAM, max. 185 MByte auf der Festplatte) bietet das Modell 60. Das Standardmodell 60 wird von einer mit 10 MHz getakteten 80286 CPU angetrieben, hat eine 40-MByte-Festplatte, ein 3½-Zoll-Laufwerk mit 1,44 MByte und kostet etwa 12 500 DM; eine arbeitsfähige Konfiguration mit Farbbildschirm und DOS 3.3 kommt auf 17 650 DM. Über sieben 16-Bit-Steckplätze steht dem weiteren Ausbau als Abteilungsrechner nichts im Wege. Lieferbar soll das gute Stück ab Ende Juni sein.

Für besonders speicherintensive Aufgaben bietet IBM ein optisches Laufwerk an, das 200 MByte auf einer Platte speichern kann. Das Gerät, das einmal beschreibbare optische Platten verwendet, kostet 7400 DM.

Auf dem Weg zum Mainframe

Das leistungsstärkste System der neuen Familie ist das Modell 80, das mit einer 16-MHz-CPU 80386 arbeitet und schon Ende Oktober für 17852 DM zu haben sein wird. Das Modell 80/111 ist mit seiner 20-MHz-CPU 80386 noch etwas schneller, kostet 22630 DM und soll zum Ende des Jahres verfügbar sein. Außer in der preiswertesten Variante (1 MByte RAM) stehen dem Anwender 2 MByte RAM zur Verfügung. An Festplatten bietet sich ein Spektrum von 40 MByte (80-041), 70 MByte (80-071) und 115 MByte (Version 80-111). Alle Versionen haben ein 3½-Zoll-Laufwerk (1,44 MByte); ein zweites Laufwerk kann nachgerüstet werden. Die maximale Festplatten-Speicherkapazität für ein Modell der Reihe 80 ergibt sich mit 230 MByte. Über sieben Steckplätze, von denen vier 16 Bit breit und drei 32 Bit breit sind, kann man weitere Adapterkarten einsetzen. Kaum zu glauben aber wahr: auch zu diesem Modell gibt es das Basic gratis dazu. Der Mikro-Kanal ist bei diesem Rechner 32 Bit breit.

Mit den zahlreichen neuen Modellen der Personal System /2 Serie macht IBM mit aller zu Verfügung stehenden Macht deutlich, daß man in den Entwicklungsabteilungen in den letzten Jahren, in denen mit dem PC ein Standard gesetzt wurde, nicht geschlafen hat. Die Aufzählung der technischen Daten und der Leistungsmerkmale der neuen Betriebssysteme ist beeindruckend. Ob die neuen Modelle in der Praxis auch halten, was die Datenblätter versprechen und die Demonstrationsgeräte Glauben machen wollen, werden wir testen. Mit den ersten Testberichten wird es allerdings noch etwas dauern, da die meisten Systeme noch nicht verfügbar sind.

Die neuen Rechner sind in SMT aufgebaut und enthalten einige Gate-Arrays, die Nachbauern sicher einige Schwierigkeiten machen werden. Ob es den Kopierern überhaupt sinnvoll erscheint, diese Arbeit auf sich zu nehmen, scheint angesichts der für IBM-Verhältnisse niedrigen Preise unwahrscheinlich. Bei den Festplatten werden Angaben von IBM zufolge durchschnittliche Zugriffszeiten von 40 ms (85 ms beim Modell 30) erreicht. Der Interleave-Faktor konnte beim 115 MByte-Laufwerk auf 1:1 (2:1 bei den Modellen 60 und 80) gesenkt werden. Den zahlreichen Grafik-„Standards" fügte IBM einen weiteren, den VGA hinzu. Damit kann man 262 144 Farben erzeugen und in einer Auflösung von maximal 1024 × 768 Punkten arbeiten. Mit dem Betriebssystem /2, das in etwa einem Jahr verfügbar sein soll, wird auch für anspruchsvolle Anwender eine Lösung von Speicherplatzproblemen angeboten werden. Wie die grafische Benutzerführung im Detail aussehen soll, konnte noch nicht in Erfahrung gebracht werden. In einer Erklärung von IBM heißt es: Das Betriebssystem /2 ist der Wachstumspfad für PC-DOS ...

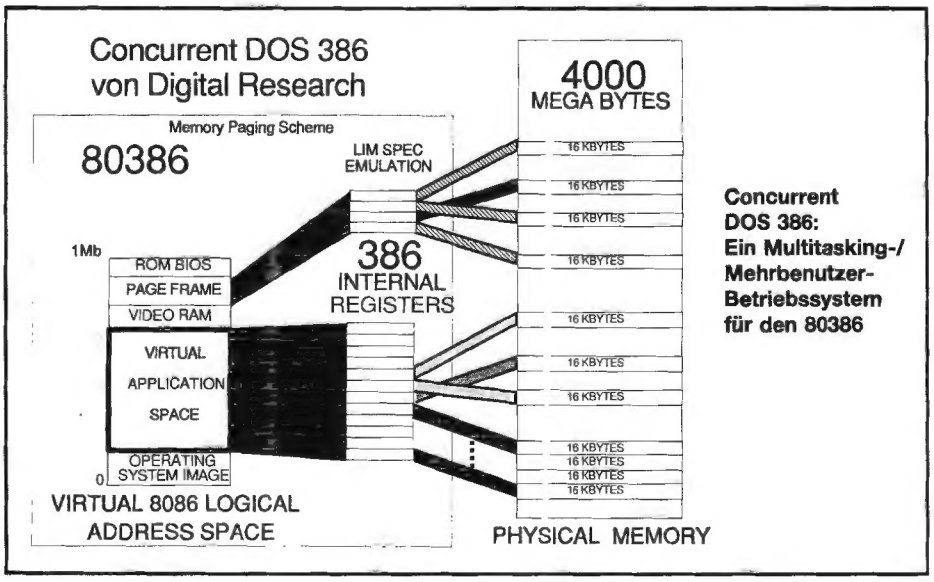
Bis zu 16 MByte RAM kann in den neuen Modellen direkt angesprochen werden. Großes und erklärtes Ziel der IBM ist es, dem Personal System /2 den Anschluß an die IBM-Großrechner zu vereinfachen. Dazu hat IBM die Systemanwendungsarchitektur SAA definiert, die ein hohes Maß an Kompatibilität zwischen IBM-Rechnern aller Größen garantieren soll.

80386-Einsteckkarte für den AT

Alle AT-Anwender können mit einer neuen Einsteckkarte von Intel, dem Inboard 386/AT, aus ihrem AT eine 386er-Maschine machen. Programme laufen mit dieser Karte ungefähr 2...3,5fach schneller ab. Auf der Karte ist ein schneller Zwischenspeicher (Cache) mit 64 KByte integriert, auf den der 80386 direkt mit seinem 32-Bit-Datenbus ohne Wartezyklen zugreift. Inboard 386/AT kann mit insgesamt 3 MByte RAM bestückt werden. Die Grundversion ist mit 1 MByte RAM ausgerüstet. Um für zukünftige Betriebsysteme gewappnet zu sein, wird der Speicher als Extended Memory betrieben. Beim AT wird der Speicherbereich oberhalb von einem MByte als Extended Memory bezeichnet, da der 80286-Prozessor diesen Speicherbereich nur im Protected Mode ansprechen kann. Dagegen dient beim Expanded Memory der Speicherbereich oberhalb FFFF:FFFFH (1 MByte) als Auslagerungsbereich für derzeit nicht benötigte Daten. Zusätzlich zur Karte wird der Intel Expanded Memory Manager (I-EMM) geliefert, damit bereits von den derzeit verfügbaren DOS-Versionen bis zu 8 MByte Speicher angesprochen werden können. In Kürze will Intel eine 386-Controll-Software herausbringen, die den virtuellen 8086-Modus des 80386 verwendet, um Anwendern des Inboard 386/AT Multitasking zu ermöglichen. Sämtliche auf dem Markt erhältliche Programme ab DOS 3.0 können in dieser Multitasking-Umgebung arbeiten.

Multitasking-Betriebssystem für den 80386

Unter der Bezeichnung Concurrent DOS 386 bringt Digital Research ein Betriebssystem auf den Markt, das die Leistungsmerkmale des 80386 ausnutzt. Concurrent DOS 386 ist ein Multitasking-/Mehrbenutzer-Betriebssystem, das zu bereits vorhandener Software unter Concurrent DOS, PC-DOS, MS-DOS und CP/M-86 kompatibel ist. Anwendungen, die auf die Speichererweiterungs-Spezifikation LIM/EMS (LIM = Lotus, Intel, Microsoft; EMS = Expanded Memory Specification) zugeschnitten sind, sind lauffähig, ohne daß spezielle Speichererweiterungskarten installiert werden müssen. Concurrent DOS 386 kann den gesamten physikalischen Adreßraum des 80386 (4 GByte) ansprechen. Dazu wird der physikalische Adreßraum in Seiten zu je 16 KByte unterteilt und der 80386 im virtuellen 8086-Modus betrieben. Die im 80386 integrierte Speicherverwaltungseinheit übernimmt die Umsetzung der logischen in physikalische Adressen, so daß die im ganzen Speicher verstreuten Seiten als Teil eines Adreßraumes von 1 MByte erscheinen. Da vom Betriebssystem keine Daten innerhalb des Speichers umkopiert werden müssen, findet die Programmausführung und die Taskumschaltung bei voller Prozessorgeschwindigkeit statt. Bis zu 255 Tasks und acht Terminals können von Concurrent DOS 386 verwaltet werden. Jede Anwendung hat ein eigenes Fenster, welches in seiner Größe, Farbe und Bildschirmposition vom Anwender bestimmt werden kann. Zwischen Anwendungen können Nachrichten ausgetauscht werden. Concurrent DOS 386 ist laut Hersteller kompatibel zu den weitverbreiteten DOS 2.X-Anwendungen wie Lotus 1-2-3, Framework, dBase und Symphony. GEM-Anwendungen werden ebenso unterstützt. Wichtig ist, daß die Diskettenformate von PC-DOS, MS-DOS und CP/M-86 erkannt werden. Damit Concurrent DOS 386 läuft, wird ein 386er-PC mit mindestens 512 KByte Hauptspeicher (Einplatzbetrieb) benötigt. Grafik-Karten, die dem CGA- und EGA-Standard entsprechen, werden vom Betriebssystem unterstützt. Laut Digital Research wird im zweiten Quartal 1987 eine Endanwender-Version von Concurrent DOS 386 erhältlich sein.



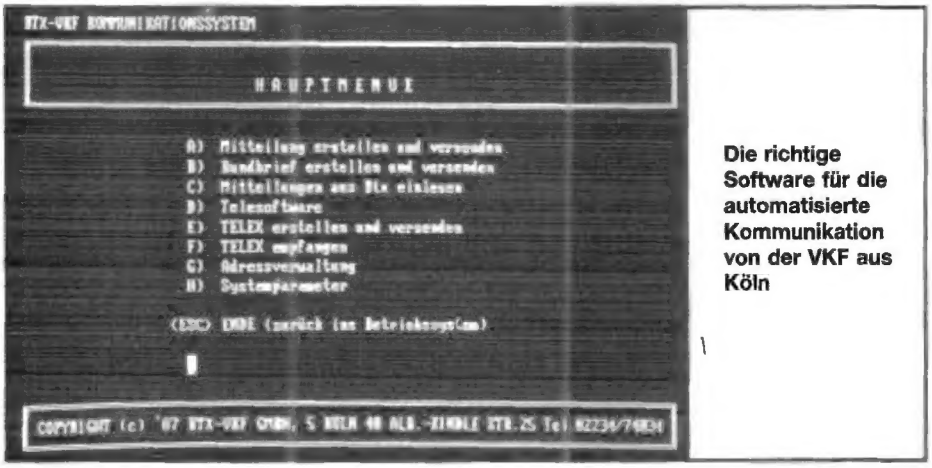
Concurrent DOS 386: Ein Multitasking-/Mehrbenutzer-Betriebssystem für den 80386

Microsoft kündigt multitaskingfähiges Betriebssystem an

Mit MS-OS/2 hat Microsoft ein multitaskingfähiges Betriebssystem für PCs, die mit den Prozessoren 80286 und 80386 ausgestattet sind, angekündigt. Das neue Betriebssystem ist das erste Ergebnis des gemeinsamen Entwicklungsabkommens zwischen IBM und Microsoft. MS-OS/2 ist wie MS-DOS ein OEM-Betriebssystem, d. h., daß es von den PC-Herstellern auf ihre spezielle Hardware angepaßt werden muß und auch nur von diesen beziehbar ist. Die existierende MS-DOS-Applikationssoftware soll laut Hersteller auch unter MS-OS/2 lauffähig sein. Der Hauptspeicher darf bis zu 16 MByte groß sein und bis maximal 1 Gigabyte virtueller Speicher können verwaltet werden. MS-OS/2 unterstützt sowohl den Real- als auch den Protected-Modus der Prozessoren 80286 und 80386. Neben dem Betriebssystemkern besteht MS-OS/2 aus dem „Windows-Presentation-Manager", der nicht anderes als eine grafische Bedienungsschnittstelle darstellt. Ein wesentliches Merkmal ist die prioritätengesteuerte Task-Verwaltung (preemtive scheduling). Ein neues Applikationsinterface macht Applikationen – unabhängig von Hardware-Unterschieden – auf verschiedenen Computern lauffähig. Das direkte Ansprechen von BIOS-Routinen, das die Ursache vieler Inkompatibilitäten ist, soll damit der Vergangenheit angehören.

Um MS-OS/2 einsetzen zu können, sollte der PC mit mindestens 1,5 MByte Hauptspeicher und einer Festplatte ausgestattet sein. Der Window-Presentation-Manager setzt einen Grafikadapter voraus. MS-OS/2 wird im 4. Quartal 1987 an die PC-Hersteller ausgeliefert. Laut Microsoft wird dieses Betriebssystem fast auf jedem 286/386-PC lauffähig sein. Für Entwickler von MS-OS/2-Anwenderprogrammen wird ab August dieses Jahres ein Software-Entwicklungs-Kit (SDK) angeboten, der eine Vorabversion des Betriebssystemskerns, technische Spezifikationen für den Windows-Presentation-Manager und den LAN-Manager, eine neue Version des Microsoft-Macro-Assemblers (MASM), einen C-Compiler, einen Programm-Editor sowie den „CodeView"-Debugger enthält. In der Lizenzgebühr sind ein Kurzseminar und regelmäßige Up-Dates der Vorabversionen enthalten.

Für MS-OS/2 hat Microsoft einen LAN-Manager angekündigt, unter dessen Regie ein PC gleichzeitig als Arbeitsstation und als Server fungieren kann. Der LAN-Manager bietet transparentes Datei- und Drucker-Sharing, verbesserte Fehlerschutz-Möglichkeiten und leistungsfähige Netzwerk-Administrations-Werkzeuge. Er ist im MS-OS/2-Betriebssystem integriert. Die Interprozeß-Kommunikations-Schnittstelle läßt es zu, daß zukünftige Anwenderprogramme direkt untereinander kommunizieren können, obwohl sie auf verschiedenen Maschinen innerhalb des Netzwerkes resident sind. Diese neue Netzwerksoftware wird im ersten Quartal 1988 erhältlich sein.



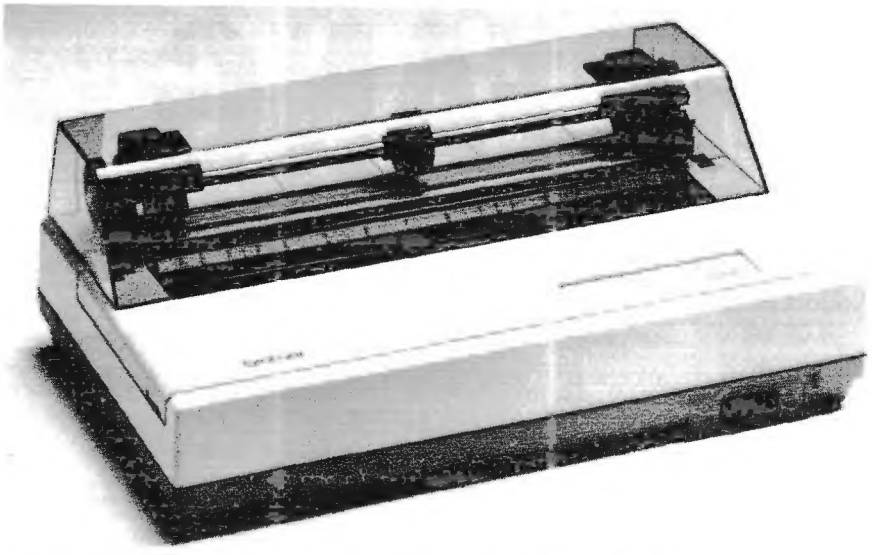
Die richtige Software für die automatisierte Kommunikation von der VKF aus Köln

Preiswert ans Telex-Netz

Die Deutsche Bundespost bietet Btx-Anwendern jetzt Zugang zum weltweiten Telex-Netz. Ein Rechner, der in Düsseldorf steht, stellt die Schnittstelle zwischen den beiden Kommunikationsarten dar. Über ihn läuft der gesamte Datenverkehr zwischen Btx- und Telex-Anschluß. Jedes Telex, das zu einem Btx-Teilnehmer geschickt wird, speichert der Rechner in einer Mailbox ab, die dann zu beliebigen Zeiten gelesen werden kann.
Was braucht nun der normale PC-Besitzer, um von dieser Neuigkeit profitieren zu können – und vor allem, was kostet es? Zunächst einmal muß er einen Btx-Anschluß beantragen. Kosten: 65 DM einmalig, 8 DM monatlich. Dafür erhält er von der Post ein Btx-Modem, mit dem er zum Nahtarif mit dem Düsseldorfer Rechner kommunizieren kann.
An Hardware ist ein Btx-Decoder erforderlich, wie er von der Firma Rafi z. B. für rund 1140 DM angeboten wird. Zusätzlich empfiehlt sich Software, die den Betrieb komfortabel gestaltet. Ein solches Programm gibt es von der Btx Verkaufsförderungsgesellschaft mbH, Köln, für rund 700 DM.

Zwei Neue von Brother

Um zwei Matrixdrucker hat die Firma Brother ihr Produktspektrum erweitert. Das Modell M-1109AP wurde speziell für den Apple-IIc entwickelt. Es bietet unterschiedliche Schriftarten und -größen und erreicht eine Geschwindigkeit von 90 Zeichen/s. Neben Schönschrift („annähernd Briefqualität") kann dieser Drucker auch Grafiken zu Papier bringen. Im Preis von 849 DM ist das Anschlußkabel für den Apple bereits enthalten.
Etwas mehr, nämlich knapp 4500 DM, muß man für den 18-Nadel-Drucker M-4018 ausgeben. Er bringt es auf beachtliche 480 Zeichen/s und schafft in seiner schönsten Schriftart, für die sich der Druckkopf dreimal über das Papier bewegt, 67 Zeichen/s. Ein Puffer von 32 KByte ist eingebaut.



Speziell für den Apple-IIc wurde dieser Drucker von Brother entwickelt

32-Bit-Prozessor mit 17 MIPS

Das Münchener Unternehmen Advanced Micro Devices hat angekündigt, einen besonders schnellen 32-Bit-Mikroprozessor zu produzieren. Der Am29000 soll 17 Millionen Befehle in der Sekunde, unter bestimmten Voraussetzungen gar 25 Millionen, abarbeiten können. Durch eine rationalisierte RISC-Architektur, die besonders auf optimierende Compiler hin ausgelegt wurde, sollen diese Leistungswerte erreicht werden. In den Prozessor mit 192 Mehrzweckregistern ist eine MMU eingebaut, die zur Adressumsetzung nur einen Taktzyklus benötigt.
Wie der Am29000 ist auch der Arithmetik-Coprozessor Am29027 in einem Pin Grid Array mit 169 Anschlüßen untergebracht. Die Serienfertigung soll im ersten Halbjahr 1988 anlaufen.

1-MBit-RAMs aus Deutschland

Im November 1985 begann die japanische Firma Toshiba als erstes Unternehmen mit der Serienfertigung von dynamischen 1-MBit-RAMs. Heute stellt sie etwa 1 Mio. Stück im Monat her. Dazu trägt seit kurzem auch Toshibas Braunschweiger Fertigungsstätte bei (1982 hatte der Halbleiterhersteller dort seine deutsche GmbH gegründet).
Der japanische Konzern kooperiert seit 1985 auch mit der Firma Siemens auf dem Speichersektor. Ziel ist es, daß beide Hersteller künftig gegenseitig als Zweithersteller für 1-MBit-RAMs auftreten. Siemens fertigt die Bausteine zur Zeit allerdings noch nicht.
Neu im Geschäft mit 1-MBit-RAMs ist Panasonic, ein weiterer Japaner. Das Unternehmen ging kürzlich mit mehreren Versionen in die Produktion: Der MN411000-10 hat eine maximale Zugriffszeit von 100 ns und eine minimale Zykluszeit von 190 ns. Er nimmt 440 mW auf und kommt im Standby-Betrieb mit 27,5 mW aus. Eine 120-ns-Version ist ebenfalls erhältlich.

OBJEKTORIENTIERTE PC-UMGEBUNGEN

Objektorientierte Entwicklungsumgebungen unterstützen die Implementation komplexer Expertensysteme für Anwendungen aus den Bereichen Prognose, Simulation, etc.. Ausgefeilte Möglichkeiten zur flexiblen Repräsentation von Wissensstrukturen und Schnittstellen zur Einbettung in Ihre bestehenden Applikationen bieten den notwendigen Komfort bei Entwicklung und Wartung von Expertensystemen.

Messenger

Objektorientierte Entwicklungsumgebung für Golden Common Lisp-Developer 286.

C-Talks

Vollständige objektorientierte C-Entwicklungsumgebung mit Bit-Map-Editor, Mausunterstützung, etc..

C++

Objektorientierter Precompiler für die Sprache „C", unterstützt die Microsoft-Implementation.

Frame Engine

Mächtiges AI-Tool, basierend auf der Sprache KRL, implementiert in Prolog 2.

Personal Consultant plus

Das bekannte in Scheme-Lisp geschriebene Entwicklungswerkzeug von Texas Instruments.

Wir beraten Sie gerne bei der Auswahl der geeigneten Werkzeuge für die Implementation leistungsfähiger Anwendungen der Wissensverarbeitung in Ihrer spezifischen Wissensdomäne.
Fordern Sie detaillierte Informationen an:

Brainware
Ihr Experte in Expertensystemen
Consulting · Schulung · Software

Brainware GmbH · Gustav-Meyer-Allee 25
1000 Berlin 65 · Tel. 030/4694696 oder
Tel. 030/4633048 · Teletex 308145-BIG

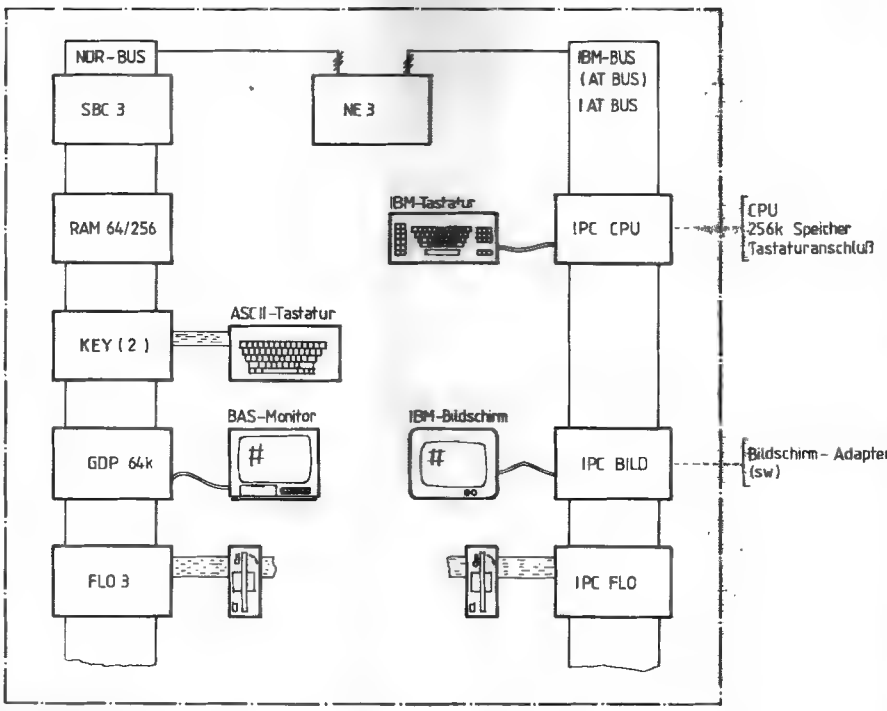
mc-info

Weniger Strahlen

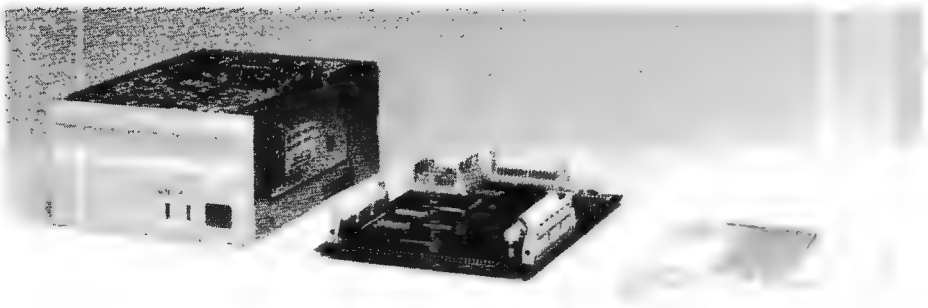
Mit einem neu entwickelten Magnetspulen- und Erdungssystem ist es Nokia Informations Systems gelungen, die vom Bildschirm ausgehenden magnetischen und elektrischen Feldstrahlen nahezu vollständig zu eliminieren. Die magnetische Abstrahlung in einem Abstand von 30 cm vom Bildschirm wird auf 20 Milli-tesla pro Sekunde reduziert. Das entspricht etwa 10 Prozent der bislang gemessenen Werte. Durch Erdung der Bildschirmoberfläche werden zusätzlich die elektrischen Felder aufgehoben. Internationale Untersuchungen haben ergeben, daß Bildschirmstrahlen für Benutzer ungefährlich sind. Zur „Beruhigung" der Anwender bietet Nokia die nötigen Komponenten an, die auch in alte Bildschirme problemlos und kostengünstig eingebaut werden können.

PC bedient NDR- und ECB-Baugruppen

Viele Mikrocomputer-Enthusiasten haben sich in mühsamer Kleinarbeit den NDR-Klein-Computer oder den mc-CP/M-Computer zusammengebaut und Stück für Stück erweitert. Dann wurden sie von der IBM-PC-Welle überrollt. So mancher stellte daraufhin das gute Stück in die Ecke und legte sich einen „Kompatiblen" zu. Für alle Besitzer der genannten Modelle gibt es jetzt eine gute Nachricht: Die Firma Graf aus Kempten bietet seit kurzem Baugruppen an, die den PC-Bus mit dem NDR- bzw. ECB-Bus verbinden.
Grundgedanke dieser Entwicklung ist, daß die Peripheriekarten des bisherigen Rechners vom PC aus zu bedienen sind. Einzige Einschränkung: Alle Baugruppen müssen DMA-fähig sein, d.h. sie müssen ihren Bus in den Tri-State-Zustand schalten können.
Wer noch keinen PC besitzt, kann eine geeignete Mutterplatine mit AT-Steckplätzen von der Firma Graf erwerben, die er wahlweise als normalen PC oder als AT betreiben kann. Entsprechende CPU-Einsteckkarten stehen zur Verfügung. Die ganze Einheit läßt sich im Gehäuse des NDR-Klein-Computers unterbringen und von dessen Netzteil versorgen. Tastatur, Disketten- und Festplatten-Controller müssen für beide Rechner gesondert vorhanden sein.



Zwei Rechner in einem Gehäuse: In den NDR-Klein-Computer läßt sich neuerdings ein kompletter PC einbauen



Zum Archivieren von Daten eignet sich dieses optische Plattenlaufwerk; der Datenträger läßt sich nur einmal beschreiben

600 MByte auf optischer Platte

Optische Speicher für den Anschluß an Computer mit SCSI-Interface bietet die Firma Hitachi an. Seit kurzem gibt es von diesem Unternehmen Laufwerke für 5¼-Zoll-Platten. Eines der neuen Modelle hat eine Bauhöhe von 83 mm, ein weiteres ist 35 mm hoch (Slimline). Sie bringen auf jeder Seite der Platte 300 MByte an Daten unter. Die mittlere Zugriffszeit beträgt etwas mehr als 100 ms, als Fehlerrate gibt der Hersteller einen Wert von 10^{-12} (mit Fehlerkorrektur) an.

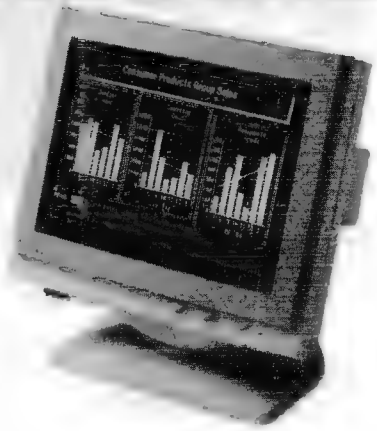
24-Nadel-Drucker paßt an Commodore-Computer

Nur knapp 1300 DM kostet der neue 24-Nadel-Drucker SL-80VC von Seikosha. Das Gerät bringt maximal 135 Zeichen/s und in Briefqualität 54 Zeichen/s zu Papier.
Neben dem Papiertransport mit Andruckrolle und Stachelwalze steht als Zusatz ein Einzelblatteinzug zur Verfügung.

Der Drucker ist speziell für den Anschluß an die Commodore-Computer C-64 und C-128 ausgelegt. Seine Steuerzeichen entsprechen denen des Commodore-Modells MPS-801. Als Druckattribute stehen zur Verfügung: Fettdruck, Sperrdruck, Hoch- und Tiefstellen, Doppelanschlag und Unterstreichen.

Wieder einmal setzt Seikosha auf einen niedrigen Preis: Der neue 24-Nadel-Drucker paßt an die Commodore-Computer C-64 und C-128

VISA FM1400



EIN NEUER MONITOR STELLT SICH VOR

FLATSCREEN,
DIE NEUE DIMENSION DER MONITORE.

Superflacher Bildschirm, ultrahohe Auflösung eröffnen dem Anwender neue Perspektiven in Grafik und Text. Helligkeits- Kontrastregler sowie AN/AUS-Schalter befinden sich auf der Frontseite.

Wählen Sie zwischen amber, grünem und weißem Bildschirm.

VISA FM 1400, der Partner für Ihren IBM-Personalcomputer XT / AT oder kompatiblen.

Technische Daten:

* 14 Zoll Monitor
* superflacher Bildschirm
* entspiegelte Bildröhre
* Ultrahochauflösung: über 1000 Pixels
* Frequenz: horizontal 18,432 Hz
  vertikal 47 – 63 Hz
* TTL- Signale

DM 679,–

Eine neue Monitor-Generation sucht zuverlässige, engagierte Händler.

Bitte wenden Sie sich an unsere Vertriebsleitung.

Die Vertriebsrechte für VISA-Monitor-Produkte liegen exklusiv für West-Germany by

KOGA

Computer GmbH
Hanauer Landstr. 439
6000 Frankfurt/M. 1
Telefon: 0 69/41 92 40

*IBM ist ein eingetragenes Warenzeichen der International Business Machines Corp.

Taschenrechner kann auf Zahlen verzichten

Die meisten haben das Kapitel Taschenrechner wohl schon abgehakt. Denn welche Fortschritte könnte man auf diesem Gebiet noch erwarten, wo doch die programmierbaren Vertreter dieser Gattung im Prinzip ihren großen Brüdern, den Tischcomputer, entsprechen? Der Firma Hewlett-Packard blieb es wieder einmal vorbehalten, die Fachwelt mit einem Knalleffekt zu überraschen.
HP-28C nennt sich das neue Wunderding, das nach Ansicht des Herstellers den Beginn der dritten Taschenrechnergeneration repräsentiert. Das entscheidend Neue daran: Mathematische Berechnungen (und ausschließlich dafür wurde das Gerät entwickelt) werden nicht mehr nur numerisch, sondern auch symbolisch durchgeführt. Das heißt, symbolischen Variablen wird nicht mehr sofort ein Wert zugewiesen, sondern das Ergebnis einer Addition kann sich auch aus A + B ergeben. Das klingt nicht überwältigend, aber umgelegt auf die Differential- und Integral-Rechnung ist es eben doch eine erstaunliche Leistung. Vergleichbar ist der neue Taschenrechner mit Programmen, die nach den Regeln der symbolischen Mathematik arbeiten. Ein Beispiel: Mit dem Gleichungslöser des HP-28C lassen sich anwenderspezifische Gleichungen eingeben und speichern. Anschließend kann der Rechner eine Gleichung nach jeder Unbekannten auflösen und deren Wert als Ergebnis ausgeben.

Auch Vektor- und Matrizen-Funktionen sind für den HP-28C kein Problem; das Gerät kostet rund 600 DM

24-Nadel-Drucker trotzt hartnäckiger Software

Eigentlich sollte es ja die einfachste Sache der Welt sein, einen Drucker am Computer zu betreiben. Aber manches Mal hat man doch seine Probleme, z.B. wenn ein Programm partout von sich aus Schrifttyp und Seitenlänge des Papiers einstellen will. Für diesen Fall hat der neue Star-Drucker NB24-10 vorgesorgt: Er läßt sich auf eine Betriebsart einstellen, in der diese Parameter von keinem Programm mehr verändert werden können. Doch das ist nur ein Beispiel für die vielfältigen Funktionen, die das neue Gerät bietet.

Der NB24-10 kann Papier bis zu 10 Zoll Breite verarbeiten; das Schwestermodell NB24-15 ist für 15 Zoll breites Papier geeignet. Bei normaler Druckqualität erreicht er eine Geschwindigkeit von 216 Zeichen/s. Korrespondenzqualität bringt er mit 72 Zeichen/s zu Papier. Heutzutage selbstverständlich: Der IBM-Zeichensatz ist fest eingebaut. Neben der in ROMs gespeicherten Schrift bietet der Hersteller drei Einschubmodule für die Schriftarten Courier, Orator und Letter Gothic an. Sämtliche Schriftarten lassen sich auch kursiv oder als Proportionalschrift darstellen.
In der Standardausführung enthält das Gerät einen Puffer von 5 KByte. Mit RAM-Modulen, die sich auch als Zeichengenerator für selbst definierte Zeichensätze nutzen lassen, kann man die Pufferkapazität auf insgesamt 85 KByte ausbauen. Bemerkenswert ist auch die Grafikauflösung, die beim Modell NB24-10 2880 Punkte/Zeile beträgt. Insgesamt sind elf verschiedene Grafikbetriebsarten wählbar. In der Standardausführung enthält der Drucker eine Centronics-Schnittstelle.

Zwei neue 24-Nadel-Drucker gibt es jetzt von Star: Hier der NB24-15, der gegenüber dem kleineren Modell breiteres Papier verarbeiten kann

Modelleisenbahn im Unterricht

Am schnellsten begreift man eine Sache, wenn man sie interessant findet. Diese Überlegung hat wohl bei den über 30 Schulen in Niedersachsen, Baden-Württemberg und Rheinland-Pfalz eine entscheidende Rolle gespielt, als sie das Märklin-Programm „train-ing" im Unterricht einsetzten. Die Hardware dieses Programms besteht aus einer elektrischen Modelleisenbahn, die über eine serielle Schnittstelle an verschiedene Computer angeschlossen werden kann. Daneben liefert Märklin aber noch zwei weitere Komponenten, die das System erst für den Unterricht geeignet machen: Software und sogenannte Teachware.

Die Software besteht aus Testprogrammen zur Anpassung an den jeweils eingesetzten Rechner und die verwendete Computersprache sowie aus Musterlösungen zu den Lehreinheiten. Die Teachware umfaßt neben Anlagenplänen, Aufgaben und Programmlistings auch didaktische Empfehlungen und Handreichungen für die Lehrer. Um den unterschiedlichen Lehrinhalten gerecht zu werden, wird zum Basispaket zielgruppenspezifische Teach- und Software angeboten. Einer der Schwerpunkte ist z. B. „Regeln und Steuern".

Unterricht, der Spaß macht: Eine Modelleisenbahn demonstriert im Unterricht zahlreiche Einsatzgebiete des Computers

Die NEUEN PLOTTER von GRAPHTEC

GRAPHTEC, weltweit bekannter Hersteller von Digital-Plottern, Analog-Recordern und Digitalisiertabletts, stellt eine Reihe neuer Plotter vor:

GRAPHTEC 24 MONATE GARANTIE

Serie PD 9011 und PD 9111
Großformatige (A0 und A1) Multifunktions-Plotter

Serie PD 9311F und PD 9411F
Griprollen-Plotter (A3 und A4) mit Einzelblatteinzug

Plotter MS 8603
Rastercorder mit Thermokamm ermöglicht A3-Plotts in 30 Sekunden oder 500 Buchstaben/Sekunde

Serie 3000
Low-Cost Plotterserie mit vielen intelligenten Befehlen.

GRAPHTEC bietet vom A4 bis A0-Plotter, in Flachbett-, Rollen- oder Trommelbauweise, ein komplettes Programm.

Fragen Sie uns – wenn es um Plotter geht.

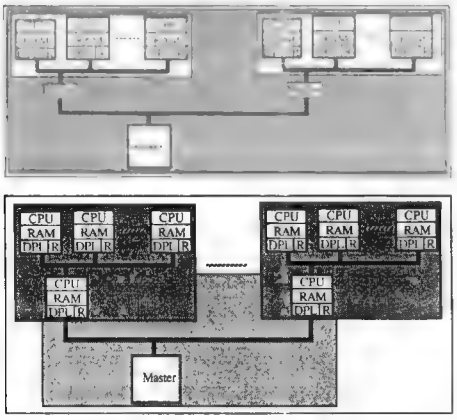
35 JAHRE SCHREIBTECHNIK

GRAPHTEC

GRAPHTEC · Postfach 1132 · D-8031 Seefeld (Obb.) 1
Berlin (030) 8835063 · Hamburg (040) 459517 · Herrsching (08152) 3084 · Düsseldorf (0211) 721145 Karlsruhe (07243) 90666 · Wien (0222) 973397

2048 68000-Rechner parallelisieren

Parwell heißt eine Computerentwicklung der p1-GmbH, zusammen mit Rolf-Dieter Klein. Dieses Computerkonzept folgt dem Master-Slave-Prinzip, bei dem sehr viele Slave-CPUs von einem Master dirigiert werden. Die Struktur des Gesamtsystems kann sich unterhalb des Masters baumartig verzweigen, wobei die Knotenrechner die Master der nächsten Ebene werden. Programmiert wird die Sache in Modula 2, wobei eine spezielle Library den Benutzer bei der Programmierung unterstützt. Vor allem die Kommunikation der Knotenrechner untereinander wird dadurch automatisiert, wodurch die gewählte Topologie des Rechners für den Benutzer transparent wird. Die Anwendung des Parwell-Rechnersystems beschränkt sich nicht auf homogene Probleme. Gerade auch die Lösung inhomogener Probleme kann mit solch einem System linerar mit der Anzahl der Slaves beschleunigt werden. Branch and Bound sowie Ray Tracing können damit in vielen Fällen im überschaubaren Zeitrahmen ablaufen.



Parwell, ein flexibles Parallel-Rechner-Konzept von p1 aus München

Nützliches für Turbo-Freunde

Zwei neue Disketten mit allerlei nützlichen Turbo-Pascal-Programmen (für MS-DOS-Computer) gibt es jetzt von Franzis-Software. Von sämtlichen Routinen liegt der Quelltext vor. Einschließlich Handbuch kostet jedes der neuen Produkte 98 DM. Turbo-Windows&Mouse bietet zahlreiche Utilities für die Fenstertechnik und die Einbindung einer Microsoft-kompatiblen Maus. Die Routinen sorgen dafür, daß beim Öffnen eines Fensters der Hintergrund gesichert wird und nach dem Schließen wieder auf dem Bildschirm erscheint. In einem Fenster kann man mit den normalen Turbo-Ein-/Ausgabe-Funktionen arbeiten. Zusätzlich gibt es fertige Funktionen für Hilfsfenster, Pop-up-Menüs und anderes.
Die Maus-Bibliothek setzt den Maustreiber von Microsoft voraus und bedient alle Funktionen des Interrupt 51.

Die Bedienung der Diskettenlaufwerke von Turbo-Pascal aus erleichtert die Turbo-Disk-Tools. Sie enthält Routinen, mit denen man z. B. Dateiattribute ansehen und ändern, Dateien in andere Verzeichnisse verschieben oder ein Gesamtverzeichnis über alle Directories hinweg erstellen kann. Die Belegung der Disketten wird grafisch angezeigt.
Außerdem erhält man auf dieser Diskette einen kompletten Disketten-Editor, der auch in übersetzter Form vorliegt; Turbo-Pascal ist in diesem Fall also nicht erforderlich.

LED-Kamm druckt 8 Seiten pro Minute

Von der Leistung und seinen Eigenschaften her entspricht NECs neuestes Druckermodell LC-08 den modernen Laserdruckern, die zu Preisen zwischen 5 000 und 10 000 DM angeboten werden. In der Technik allerdings unterscheidet es sich von ihnen: Es benutzt nicht, wie die meisten Konkurrenzprodukte, die Lasereinheit von Canon, sondern erzeugt das Druckmuster mit einem Schreibkopf, der aus 2400 Leuchtdioden besteht. Entwickelt wurde diese Einheit von NEC selbst. Die Druckgeschwindigkeit beträgt konstant 8 Seiten/min., unabhängig davon, wieviel Text oder Grafik zu Papier zu bringen ist.

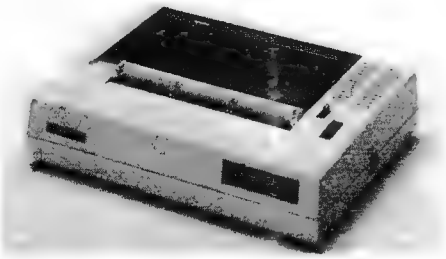
Das Gerät ist insbesondere für Desktop-Publishing interessant, das jetzt groß im Kommen ist. Der Einzelblatteinzug des LC-08 faßt 250 DIN-A4-Blätter und läßt sich durch einen zweiten Schacht mit gleicher Kapazität ergänzen. Auch Projektorfolien lassen sich mit ihm verarbeiten. Neben den vier Standardschriften stehen weitere Schriftarten als zusteckbare Kassetten zur Verfügung.

Ein sogenanntes Grafik-Board, das einen Pufferspeicher von 1,3 MByte enthält, erweitert den LC-08 zum HP-GL-kompatiblen Plotter mit einer Auflösung von 300 x 300 Punkten/Zoll.

Entspricht in seinen Leistungsmerkmalen modernen Laserdruckern: der LC-08 von NEC

Drucker en masse

Von Mannesmann Tally wurden auf der CeBIT zahlreiche neue Drucker vorgestellt. Von Matrix- über Tintenstrahl-, Nadel- und Laserdrucker spannt sich der Bogen der Neuvorstellungen. Ein 24-Düsendruckkopf mit einer garantierten Lebensdauer von 200 Millionen Zeichen sorgt im Tintenstrahldrucker MT 90 mit



Für 6 Millionen Zeichen reicht die Tinte im MT 90

220 Zeichen/sec. für schnelle und hochwertige Ausdrucke. Bei Schönschrift werden die Zeichen in einer Matrix von 24 x 18 Punkten auf das Papier gespritzt. Zu dem Drucker, der zu den IBM-Zeichensätzen 1 und 2 kompatibel ist, gibt es neben der parallelen auch eine serielle Schnittstelle.

Der Laserdrucker Tally Page Printer MT 910 schafft 10 Seiten in der Minute und holt sich sein Papier aus zwei Einzugskassetten. Die Geräuschentwicklung erreicht mit nur 50 dB(A) einen sehr geringen Wert. Zur einfachen Bedienung ist das System mit einem 16stelligen LC-Display ausgestattet, das Hinweise in mehreren Sprachen geben kann. Eine vorbeugende Wartung soll bei diesem Modell nicht erforderlich sein. Lediglich nach 300 000 Seiten ist eine Überholung vorgesehen. Neben den vier Standard-Zeichensätzen können über Steckmodule weitere Zeichensätze eingebaut werden.

Sehr leise und durch Steckmodule äußerst flexibel ist der Laserdrucker MT 910 von Mannesman Tally

TOP Software/Hardware PC/XT/AT

Bitron-FIBU 409,-
Kred.,Deb.,OP,Mahnsystem G&V,Bilanz,10xMwst, Mwst-Anmeldung, Mandantenfähig
FIBU-Demo 50,-
FIBU-Sourcecode + Compiler 2278,-

FAKTURA + Lagerverwaltung 330,-
Faktura-Demo 50,-
Faktura-Source + Compiler 1129,-

PC-WRITE deutsch 190,-
PC-WRITE incl. Mailmerge 340,-

Bitron-TEPP 2394,-
Lagerverwaltung für den Teppich-Fachhandel
Demo-Version 100,-

Tresor 112,-
Elektronische Datei-Geheimhaltung

PageMaker 2265,-
Desk Top Publishing
Laserdrucker od. Linotronic
... auch im Netz
Vorführtermin bitte vereinbaren.

MICROSOFT.

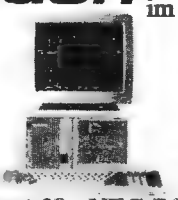
QuickBASIC	2.1	225,-
Multiplan	3.0	630,-
Windows	1.3	340,-
Word	3.1	1090,
MS-Mouse	6.0	395,-

CADD 2.0 + LogiMouse 593,-
Profi-2D-CAD,unterstützt CGA, EGA, Hercules, Tecmar, AST, Paradise, Olivetti, 8087, 80287, metr. u. engl. System, X-Y u. Polar, Bibliothek,-Text, Windows-Funktionen, Mischen, Ausschneiden, Macro u. Batch-Kommandos, Plotting auf jedem Matrixdrucker, ca. 100 Menuefunktionen.

USA Public Domain
The Best of the Best
5 Disketten (ca. 45 Programme)
Karten-, Schach-, Abenteuer-, Fightspiele, Soundbibliothek, Klaviersimulator, Flugsimulator, Tools für DOS, Basic, Drucker, Tastatur, Ramdisk.
Einzeldiskette 20,-DM Vorausscheck, bar, Briefmarken

Neu ! an alle BASIC-Programmierer
Stay-Res 297,- Demo-Diskette 20,-
Läßt Ihre Programme im Hintergrund bleiben. Die Programme können jederzeit mit Hot Key aufgerufen werden.

Tandon im NETZWERK



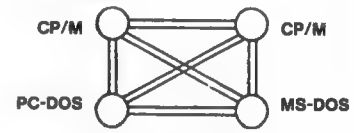
PCA-20
PCA-30
PCA-40
PCA-70

Novel 286/TS 4537,-
80287-Co./8Mhz 786,-
20 MB-Festplatte 780,-
40 MB-Festplatte 1767,-
AT-Kombi-Controller 453,-

Target-20 + NEC-P6
Komplett 7395,- kein Computerversand

UniForm (CP/M ↔ DOS) 399,-
Konvertierungsprogramm ca. 160 Formate.
Formatieren, kopieren, emulieren.

Kopier Service 5 1/4 Zoll



Pro Diskette 50,-
Neu! Apple II und Northstar Formatkonvertierung

Neu !
BREAKTHRU - 286
AT-Karte
wandelt Ihren alten XT in einen
SuperschnellenAT um
die Karte beinhaltet den
80286 Prozessor mit 8Mhz, Cache-memory,
Coprozessor 80287 optional
1490,-

Diese Anzeige ist mit PageMaker erstellt

Beim Kauf gelten Bitron-Lizenz-Bedingungen. Kein Umtausch NN + 10,- oder V.-Scheck.

Bitron GmbH
Starkenburgring 10
6057 Dietzenbach
☎ 0 60 74-2 88 38 + 2 96 45

Europäische Software-Häuser sollen kooperieren

Die Kommission der Europäischen Gemeinschaft will den Austausch von Software zwischen kleinen und mittleren Software-Häusern in Europa vorantreiben und fördern. Sie hat deshalb ein Programm verabschiedet, das die Bundesrepublik Deutschland, Großbritannien und Frankreich betrifft. Bei uns wurde die Firma GUP International Consulting GmbH (GIC), Görlitzer Str. 1, 4040 Neuß, beauftragt, alle notwendigen Maßnahmen durchzuführen.

Die Zielsetzung des Programms ist einfach: Zwischen den Software-Häusern in den drei genannten Ländern sollen Kontakte aufgebaut werden, mit dem Ziel, einen Informationsaustausch herbeizuführen. Sie sollen in der ersten Phase zur technischen Kooperation führen, um die Leistungsfähigkeit der Software zu verbessern und diese an die Bedürfnisse des europäischen Marktes anzupassen. In der zweiten Phase wird eine Kooperation in den Bereichen Vertrieb und Service angestrebt.

Die Aufgabe von GIC ist es, die interessierten Firmen mit ausländischen Partnern zusammenzubringen, die in gleichen oder ähnlichen Gebieten der Software-Entwicklung tätig sind. Eine gezielte Befragung der betreffenden Firmen stellt den ersten Schritt dar.

Flotter Japaner mit 9,54 MHz

MultiSpeed heißt NECs neuester Aktentaschencomputer. Diese Bezeichnung weist schon darauf hin, daß die japanische Firma insbesondere die hohe Geschwindigkeit des Rechners in den Vordergrund stellen will. Neben den üblichen 4,77 MHz kann der MultiSpeed nämlich auch mit der doppelten Taktfrequenz betrieben werden.
Die anderen Merkmale entsprechen denen, die bei der neuesten „Laptop"-Generation üblich sind. Das heißt: zwei 3½-Zoll-Disketten-Laufwerke (je 720 KByte), Stromversorgung über Nickel-Cadmium-Batterie oder Netz (mit automatischer Batterieladung), Flüssigkristallanzeige mit 25 Zeilen zu je 80 Zeichen oder 640 x 200 Bildpunkten, Betriebssystem – natürlich – MS-DOS, Gewicht unter 6 kg.

Der Hauptspeicher des MultiSpeed ist schon in der Standardausführung mit 640 KByte ausgestattet. Als Prozessor enthält der neue Rechner NECs den 8086-kompatiblen Typ V30. Serielle Schnittstelle, Centronics-Interface und ein RGB-Anschluß für einen externen Monitor sind ebenfalls standardmäßig vorhanden. Der neue Computer mißt 34,6 cm x 31,5 cm x 7,65 cm und kostet rund 5000 DM.

Mit 4,77 MHz oder 9,54 MHz läßt sich dieser handliche Computer von NEC betreiben

Neue Sammeldiskette

Die neueste mc-MS-DOS-Sammeldiskette (Nr. 7) kann jetzt von Franzis-Software bezogen werden. Sie enthält die Programme zu folgenden Artikeln:

– EPROM-Brenner
– Druckeranpassung unter MS-DOS
– Multitasking unter MS-DOS
– Pascal-Tracer

Alle diese Beiträge sind in Heft 1/87 erschienen. Außerdem finden sich auf der Diskette ein Programm, das Dateien in das Intel-Hex-Format umwandelt und ein weiteres, das Intel-Hex-Dateien in lauffähige Programme zurückverwandelt.

MS-DOS simuliert CP/M

So mancher hat eine Menge Geld in CP/M-Programme investiert und ist dann auf MS-DOS umgestiegen. Wer nicht zwei Computer parallel betreiben will, kann jetzt seine alten CP/M-Programme auch auf dem neuen Rechner nutzen.
Franzis-Software bietet jetzt mit CPM-80 ein Programm an, das alle Systemschnittstellen des Betriebssystems CP/M auf IBM-kompatiblen PCs simuliert. Wenn der Rechner einen der NEC-Prozessoren V20 oder V30 enthält – beide beherrschen den Befehlssatz des 8080 und des 8086 –, läuft die Simulation sogar in Echtzeit ab. Ist das nicht der Fall, werden die Befehle des 8080 softwaremäßig emuliert.
CPM-80 verwendet die Dateistruktur von MS-DOS: Unterverzeichnisse werden durch das ASSIGN-Kommando auf die CP/M-Laufwerksbezeichnungen A:...P: abgebildet. Die durch CP/M-Anwenderprogramme erzeugten Dateien lassen sich unter MS-DOS weiterverarbeiten. Zur Übertragung von CP/M-Programmen vom alten Rechner auf den PC verwendet man die serielle Schnittstelle. Entsprechende Übertragungsprozeduren liegen dem Programm als Turbo-Pascal-Quellen bei.
Die Bedienung von CPM-80 ist denkbar einfach: Das Programm enthält einen Kommando-Interpreter, der dem CCP von CP/M gleicht. Zusätzlich zu den normalen CP/M-Befehlen gibt es die Kommandos ASSIGN (Abbilden eines DOS-Unterverzeichnisses auf ein CP/M-Laufwerk) und EXIT (zurück zu MS-DOS). Das Programm kostet 298 DM.

„AT-Power“ für den PC

Für knapp 1400 DM kann man jetzt seinen IBM-PC/XT-kompatiblen Rechner in einen voll funktionsfähigen AT umwandeln. Man ersetzt dazu das Original-Motherboard des XT durch das sogenannte Baby-AT-Motherboard von Impec. Nach dieser Maßnahme bringt der Rechner die fünf- bis siebenfache Leistung eines normalen XT. Das Baby-AT-Board stellt eine vollständige AT-Hauptplatine mit dem Prozessor 80286 und einer Fassung für den Arithmetikprozessor 80287 dar. Es kann bis zu 1 MByte RAM aufnehmen und enthält das AT-BIOS. Auch eine akkugepufferte Echtzeituhr ist vorhanden. Die Taktfrequenz läßt sich zwischen 6, 8 und 10 MHz umschalten.

Zur Erweiterung mit Steckkarten stehen sechs lange und zwei kurze Slots zur Verfügung. Anschlußmöglichkeiten für Tastatur, Sicherheitsschloß und Betriebsanzeige, Lautsprecher und Hardware-Reset sind ebenfalls vorgesehen.

Wer das Baby-AT-Board in IBM-kompatible Rechner einbauen will, sollte darauf achten, daß Abmessungen, Befestigungspunkte und Lage der Anschlüsse identisch mit denen des IBM-PC-XT sind. Als Betriebssystem kommen die MS-DOS-Versionen 3.X in Frage. Das Diagnose/Setup-Programm zur Einstellung der Systemparameter und zur Systemüberprüfung wird nicht mitgeliefert.

Termine

Der Fachkongreß und die Ausstellung „AI-Europa“ findet in diesem Jahr vom 3. bis zum 5. Juni in der Kongreßhalle der Frankfurter Messe statt. Der Kongreß wird in englischer und deutscher Sprache abgehalten.

Vom 11. bis zum 14. Juni 1987 findet in Köln die C'87 statt. Über 300 Aussteller aus 20 Ländern werden zu dieser Messe erwartet, die vom PC bis zur mittleren Datentechnik einen Überblick über das aktuelle Angebot geben möchte.

Der alle drei Jahre stattfindende Weltkongreß der International Federation of Automatic Control (IFAG) wird vom 27. bis zum 31. Juli 1987 in München veranstaltet. Der Kongreß dient dem Erfahrungsaustausch auf 33 Gebieten der Automatisierungstechnik.

Die vom 29. Juni bis zum 1. Juli 1987 in der Sindelfinger Messehalle stattfindende internationale Fachmesse „SMT'87 Surface Mounted Technologies“ wird erstmals in einer eigenständigen Messe den Bereich der Oberflächentechnologien für Leiterplattenbestückung darstellen.

In der gleichen Messehalle findet vom 3. bis zum 6. Juni eine Ausstellung zum Thema Identifikations-Technologien statt.

Die Electric Indonesia findet in diesem Jahr vom 10. bis zum 14. November in Jakarta statt.

Über wissensbasierte Systeme kann man sich vom 28. bis zum 29. April 1987 in Regensburg informieren. Auf dem Symposium, das vom ostbayerischen Technologie Transfer Institut zum zweiten Mal veranstaltet wird, wird gezeigt, unter welchen Voraussetzungen Expertensysteme in einem Produktions- oder Dienstleistungsunternehmen eingesetzt werden können.

HW ELEKTRONIK Handels GmbH
2000 Hamburg 19 Eimsb. Chaussee 79
Telefon 040/439 68 48 u. 430 00 19
Ihr Elektronik-Fachgeschäft in Hamburg
EIN PREISVERGLEICH LOHNT!

TEAC-LAUFWERKE
FD 35 FN 279,—
FD 36 GF/N 359,—
FD 55 A-V 259,—
FD 55 B-V 259,—
FD 55 F-V 299,—
FD 55 GF-V 319,—
SD 510 1099,—
SD 520 1199,—
SD 540 1899,—

NEC-LAUFWERKE
FD 1063 339,—
FD 1035 LP 259,—
FD 1135C 339,—
FD 1155C 319,—
FD 1165 1189,—
FD 1035 245,—
D 5126 975,—
D 5146 799,—
D 5146 1588,—
D 5126 H 1399,—
D 5146 H 1899,—

PREH-COMMANDER
PC-IL 322,50
CD 102 EX 469,—
CD 102 P 495,—
AK 87 UNI 329,—
AK 87 (APPLE) 329,—
AK 87 (APPLE) o. Block 239,—

ERSA MS 300 115,—
ERSA MS 6000 145,—

MONITORE, 22 MHz, geëtzt
9" grün 269,—
9" orange 275,—
12" grün 279,—
dto. f. IBM/TTL 289,—
12" orange 285,—
dto. f. IBM/TTL 299,—

3M-Disketten „The Name“ 10 St.
3,5" SS/DD 57,90
3,5" DS/DD 69,80
5,25" SS/DD 29,70
5,25" DS/DD 36,90
5,25" SS/DD 96 TPI 42,90
5,25" DS/DD 96 TPI 44,50
5,25" DS/HD 96 TPI 75,90

Exp.-Karte f. „IBM-PC“ o. ä.
Epoxyd, ca. 4 k Löcher 39,50

EXP.-Karte f. „APPLE“ o. ä.
2seitig, Epoxyd 14,50

QUALITÄTS-QUARZE
32,768 kHz 0,95
1,0/1,008 MHz je 8,10
1,8432/2,0 MHz je 4,40
2,048/2,097152/
2,4576 MHz je 4,90
3,0/3,2768/3,579545 je 2,70
3,6864/4,0/4,096/4,194304/
4,433618/4,9152/5,0/
5,0688/5,12/5,185/
5,2/6,0/6,4/6,5536/
7,3728/8,0/8,877238/
9,216/10,0 MHz
10,24/12,0/14,0/14,318/
14,7456/15,0/16,0/
18,0 MHz je 1,60

QUARZ-OSZILLATOREN
4/5/6/8/10/12/14/16 MHz je 7,95

KERAM. KONDENSATOREN
RM 5 mm
100 nF/50 V % 21,—
ab 500 % 20,—
ab 1000 % 18,—

Printerkabel f. IBM o. ä.
2 m/Centr./SUB-D 17,50

EURO-BUS
10 St.-pl. 38,50
DTO, m. 10 BL 64p 65,90

IC-QUALITÄTSSOCKEL DOPPELFEDER
8—40polig p.P. 0,02

PRÄZISIONSSOCKEL vergoldet, gedreht
8—64polig p.P. 0,05

PRÄZISIONSSOCKEL WIRE-WRAP/ vergoldet, gedreht
8—40polig p.P. 0,20

DIL-STECKER
f. Flb-KABEL/anpreßb.
DIS 14 1,30 DIS 26 2,20
DIS 16 1,40 DIS 40 2,80
DIS 24 2,— DIS 20 2,10

FLOPPY-STECKER verg.
10polig 3,30
20polig 3,95
26polig 4,50
34polig 4,50
40polig 5,80
50polig 7,30
60polig 11,50

CENTRONICS-STECKER verg.
14polig/löt. 4,25
24polig/löt. 4,50
24polig/quetsch 11,50
36polig/löt. 3,25
36polig/quetsch 5,70
50polig/löt. 7,30
50polig/quetsch 21,95

CENTRONICS-BUCHSEN verg.
14polig/löt. 5,50
24polig/löt. 4,40
24polig/quetsch 16,50
24polig/90° 9,50
36polig/löt. 4,50
36polig/quetsch 5,90
36polig/90° 9,90
50polig/löt. 7,40
50polig/quetsch 23,50

CENTRONICS-KUPPLUNGEN
36polig/löt. 6,90
50polig/löt. 7,90

VG-STECKVERBINDER verg.
64p Stecker A/C 2,50
96p Stecker 3,80
64p Buchse A/C 3,75
64p Buchse 90° 8,95
64p Buchse Wire Wrap 4,30
64p Buchse/quetsch 11,50
96p Buchse 5,95
96p Buchse Wire Wrap 8,50

DIREKTSTECKER/SLOTS verg.
2x 18p RM 2,54 3,50
2x 22p dto 3,90
2x 23p dto 4,70
2x 25p dto 3,50
2x 31p dto 4,40
2x 6p RM 3,96 2,40
2x 12p dto 2,90
2x 18p dto 5,50
2x 22p dto 5,95

SUP-D-STECKVERBINDER verg.
STECKER

Pole	löt	90°	quetsch
9	1,10	2,40	5,20
15	1,50	3,50	6,50
19	3,75	—	—
23	3,90	—	—
25	1,70	5,40	7,50
37	3,50	7,90	10,90
50	4,75	10,50	34,50

BUCHSEN

9	1,25	2,90	5,50
15	1,75	4,50	6,80
19	3,90	—	—
23	4,00	—	—
25	1,80	6,60	7,70
37	3,75	9,90	12,20
50	6,90	15,50	35,50

PFOSTEN-VERBINDER 2reihig verg.; Auswerfer/Zugentl.

STECKER/90°		BUCHSEN
10p 1,40	1,85	1,20
14p 1,50	1,90	1,60
16p 1,65	1,95	1,70
20p 2,15	2,50	2,10
26p 2,50	2,95	2,30
34p 3,30	3,75	2,90
40p 4,00	4,30	3,50
50p 4,90	5,20	4,20
60p 5,90	6,10	5,50

FLACHBANDLITZE grau 14—64p
p. Ader/m —,10

PFOSTENFELD-VERBINDER
Stecker
36p Stecker 1reihig 1,60
36p Stecker dto. 90° 1,95
72p Stecker 2reihig 3,30
72p Stecker dto. 90° 3,75
36p Buchse 1reihig 3,90
36p Buchse dto. 90° 4,70
72p Buchse 2reihig 7,25
72p Buchse dto. 90° 9,25
JUMPER rt/sw 10 St. 1,80

74 LS, 74 LS, 74 LS, 74 LS.
00/02/03 je 0,45
01/04/05/08/09/10/11/12/
27/51/54/55/133/260 je 0,55
13/15/20/21/22/28/30/
32/33/37/38/136 je 0,70
14/40/73/74/75/76/78/107/
109/112/113/114/122/228/
365/366/367/368/386/ je 0,95
06/07/42/86/90/91/92/93/
125/126/132/137/138/139/
279/375 je 1,25
48/49/83/85/95/96/123/
151/153/155/156/157/158/160/
161/162/163/164/173/174/175/
193/194/195/196/197/251/253/
257/258/283/290/293/352/353/
390/393 je 1,45
47/145/169/170/190/191/
192/221/240/241/242/243/244/
247/248/249/259/273/295/298/
373/374/377/378/379/395/396/
490/540/541/640/641/642/643/
44/868/669 je 1,65
165/166/245/280/348/645 je 1,85
63/147/148/398/399/670 je 2,40
154/181/183/624/625/626 je 3,50
275/321/323 je 4,95
673/674/682/683/685/686/
89 je 6,80
686/687 je 8,30
590/591/592/595/596/597 je 8,95
593/598 je 11,50

● HW-NEWS ● HW-NEWS ●
7001C 8,50 7011C 8,50
7002C 11,50 7020B-8 109,—
7003C 21,90 7021B-8 114,—
7004C 18,50 8052AH 99,—
8087-5 299,— DAC0808 7,95
901225-01 39,90
901226-01 37,50
901227-03 37,50
901229-05 39,90
906114-01 28,50
8502 45,—
8701 25,—
8721 49,—
8722 49,—
8563 R9 189,—
8566 R3 99,—
325302-01 42,50
325572-01 42,50
TL 7705 3,—

27010-250	65,—	65SC32-2	23,50	ADC 0808	17,50
27011-250	65,—	6545	11,50	AM25LS2538	7,90
27210-250	65,—	6551	7,50	AM26LS31/2/3 je	4,10
2716-450	9,50	6551A	8,20	AY-3-1015	10,90
2732A-250	9,90	65SC51	17,30	AY-3-8910	17,75
27C32-450	15,50	65SC51-2	20,70	AY-3-8912	16,25
2764-250	7,90	6569	56,90	EF 9364	17,25
27C64-250	10,50	6581/6582 je	44,50	EF 9365/66 je	62,50
27128-200	9,50	6592	37,30	EF 9367	69,90
27256-250	11,50	6800, 02 je	7,90	FDC 9229 BT	29,90
27C256-150	15,90	6809	13,90	HD 4702	23,90
27512-250	24,50	68A09	17,50	ICL7660/8211 je	6,75
27C512-250	24,90	6810	4,30	MAX 232	14,90
4118-200	3,10	6821	3,95	MC 1488, 89 je	1,10
4164-120	2,75	6821A	4,85	MC 3242P	17,50
4164-150	2,30	6840	8,90	MC 3470N	7,50
4164-150 autom.	4,70	6845	8,50	MC 3486P	3,95
41256-120	5,60	6845B	17,90	MC 3487P	3,95
41256-150	5,30	6850	3,90	MSM 58321	8,25
41416-12	4,90	68000-8	31,50	N 82 S 23/123 je	3,95
41464-12	8,75	68HC000-8	89,—	N 82 S 126/129 je	4,95
43256-12	24,90	68000-10	37,50	NS405A12N	88,90
6116LP3	3,75	68000-12	69,50	PAL 16 L8	9,90
6116LP3	8,50	68008-8	32,50	RTC 58321	12,90
6116LFP3	8,50	68010-8	69,90	SAB 1791/3/7 je	19,90
6264LP12	8,40	68230-8	20,—	SAB 2793/97 je	24,50
6264LFP12	9,90	68483	129,—	SY 2149-2	11,50
62256LP12	25,90	68564N04	89,—	TBP 24S10/SA10 je	8,50
62256LFP12	34,50	68564N05	89,—	TBP 28L22	12,50
6303	23,90	68590	229,—	TC 511000P10	89,—
6303 E	37,50	68591	129,—	TL 497	4,50
6321	11,50	68652	28,—	TMS 4416-12	4,90
6340	15,40	68661/681 je	28,50	TMS 4464-12	8,75
6345	33,50	68020-12	549,—	TMS 4500-19	42,50
6350	11,50	68881-12	549,—	TMS 9902	19,75
63265	49,—	68901	53,—	TMS 9995	69,—
63450-8	188,—	7910/11 je	49,90	U 9216	13,20
63463-8	197,—	8035	5,90	Z 80A CPU C/M	7,90
64180-4	83,—	8039	8,50	dto. CTC/PIO je	12,50
65SC816-4	92,—	8060A	9,90	dto. SIO-O	33,90
6501Q	38,50	8085A	5,90	Z 80A CPU	3,30
6501AQ	42,50	8086	19,90	dto. PIO/CTC je	4,25
6502	7,50	8087-2	389,—	dto. DART/DMA je	9,90
6502A	8,20	8088	16,90	dto. SIO-O	10,50
65C02	15,50	80287-8	687,—	dto. STI	24,50
65SC02	14,50	8155	5,20	Z 80B CPU	6,50
65SC02-2	16,90	8212	3,90	dto. PIO/CTC je	7,70
6504	8,50	8214	8,80	dto. DART/SIO-O je	15,50
6504A	11,90	8216	3,90	dto. STI	35,30
65SC04	16,90	8224	5,15	Z 80H CPU	9,75
65SC04-2	17,90	8226	6,90	ZN 425	13,40
6510	18,50	8228	8,70	ZN 426	7,80
6511Q	38,50	8237	11,90	ZN 427	25,90
6511AQ	42,50	8243	4,50	ZN 428	18,30
6520	3,70	8250	23,50	µPD 765 AC	13,50
6520A	8,30	8251, 53	5,50	µPD 1990	8,50
65SC21	15,80	8255	4,95	µPD 70108-8 (V20)	18,90
65SC21-2	17,40	8257, 59, 79 je	5,60	µPD 70116-8 (V30)	22,50
6522	7,50	8282, 84 je	7,90	µPD 71055C	7,80
6522A	8,75	8286, 87 je	8,50	µPD 71082C	8,50
65SC22	17,70	8288	12,75	µPD 71088C	21,50
65SC22-2	18,30	8289	29,50	µPD 7220AD	36,50
6526	17,90	8748 HD	24,50	**Versand per Nachnahme ab DM 30,— + Versandkosten; Zwischenverk. vorbeh., Erf.-Ort Hmb.**	
6532	7,50	8749 HD	24,90		
6532A	11,50	8755 AD	23,90		
65SC32	21,30	ADC 0804	10,75		

NEC bringt CMOS-Chip für hochauflösende Grafik

In CMOS-Technik wird NECs neuester Grafik-Chip gefertigt. Der Baustein nennt sich Advanced-Graphic-Display-Controller (AGDC) und trägt die Typenbezeichnung µPD72120. Er verwaltet einen Bildspeicher, der praktisch beliebig groß sein kann. So bereitet es keine Probleme, ein System mit 64 Bildebenen zu je 2048 x 2048 Punkten aufzubauen. Für einen hohen Durchsatz sorgen zwei interne Prozessoren, von denen einer für die Zeichendarstellung zuständig ist.
Mit dem Host-Rechner ist der Prozessor über einen 16-Bit-Datenbus verbunden, den man auf 8 Bit umschalten kann. Der Bildspeicher wird über 20 Adreßleitungen angesprochen. Auf der Videoseite sind 32 Adreßleitungen vorhanden.

Der AGDC bietet leistungsfähige Befehle zum Plotten von Punkten, Vektoren, Polygonen, Rechtecken, Kreisen, Ellipsen, Sektoren und Bögen mit absoluten oder relativen Koordinaten. Paint-Befehle füllen beliebige Flächen mit Mustern. Für die Blockübertragung im Bit-Modus verfügt der Baustein über vielseitige Copy-Befehle, mit denen man beliebige Rechteckbereiche in andere Speicherbereiche kopieren oder beliebige Fenster erzeugen kann. Damit lassen sich z.B. verschiedene Schrifttypen (auch Proportionalschrift) erzeugen. Eine logische Verknüpfung der Quell- und Zielinhalte ist möglich. Beim Kopieren kann das Zielbild nach Wunsch gedreht, geneigt oder in der X- oder Y-Achse verkleinert bzw. vergrößert werden. Auch das Ausblenden verdeckter Bereiche beherrscht der Baustein (Clipping).
Put- und Get-Kommandos sorgen für einen schnellen Datenaustausch mit dem Host-Rechner. Der Bildspeicher kann dabei entweder als Teil des Host-Speichers betrachtet oder aber per DMA bedient werden. Der AGDC konvertiert lineare Adressen selbständig in zweidimensionale Adressen und umgekehrt. Aufgrund seiner Eigenschaften eignet sich der Chip insbesondere für hochauflösende Grafik, die mit Text kombiniert werden soll.

80386-Entwicklungssystem

Mindestens zehnmal schneller als alle herkömmlichen Entwicklungssysteme mit 16-Bit-Prozessoren ist laut Angaben der Firma Siemens deren neuestes Modell SME-V531. Die Anlage läuft mit dem 32-Bit-Prozessor 80386 und soll vom Leistungsniveau her einer professionellen Workstation entsprechen. Im Preis liegt sie mit etwa 35 000 DM aber deutlich darunter. Inbegriffen sind 640 KByte RAM, Diskettenlaufwerk (1,2 MByte), Harddisk (32 MByte), Farbmonitor und das Betriebssystem MS-DOS 3.20.

Das Gerät läßt sich leicht in eine bestehende SME-Umgebung einbinden. Die Emulatoren SME-IETA und PMS-A500 sind anschließbar. Auf der Anlage laufen auch die neuen AT-kompatiblen Software-Tools (HLL-Compiler).

Einer professionellen Workstation entspricht dieses 80386-Entwicklungssystem von Siemens

Versados unterstützt Zusatzbausteine

Das Echtzeitbetriebssystem Versados von Motorola unterstützt in seiner neuesten Version 4.5 nicht nur die CPU 68020, sondern auch den Gleitkomma-Coprozessor 68881 sowie die Paged-Memory-Management-Unit (PMMU) 68851. Zahlreiche Treiber sind jetzt für diverse Massenspeicher vorhanden: für 5,25- und 8-Zoll-Diskettenlaufwerke, Harddisks bis 70 MByte und Streamer bis 60 MByte. Außerdem läßt sich das große Angebot an VME-Bus-kompatiblen Prozessorplatinen und Schnittstellen-Modulen uneingeschränkt nutzen. RMS68K, der Echtzeitkern von Versados, kann auch für Anwendungen ohne Plattenspeicher verwendet werden.

Pocket-Guide lichtet Speicher-Dschungel

Ab sofort ist Band 4 der Pocket-Guide-Serie von Texas Instruments verfügbar. Er enthält in übersichtlicher Form Kurzbeschreibung und wesentliche Kenndaten der integrierten Speicherbausteine dieses Herstellers. Pocket-Guides sind keine Datenbücher im herkömmlichen Sinn, sie sollen den Entwickler bei der Auswahl des richtigen Bauelements unterstützen. Alle Bücher dieser Serie erklären deshalb Begriffe und Funktionen sehr ausführlich. Eine detaillierte Kapitelübersicht am Anfang der Bücher erleichtert dem Anwender das schnelle Auffinden der gesuchten Typen. Der jetzt vorliegende vierte Band enthält bipolare RAMs, PROMs und PALs, EPROMs, OTPs, EEPROMs, DRAMs, SIPs und integrierte DRAM-Controller.

Programmierung

Prof. Dr. Franz Stetter, Dr.-Ing. Helmut Balzert. Programmierung, Reihe Informatik, Band 50, Kartoniert, Bibliographisches Institut Zürich, BI-Wissenschaftsverlag, 373 Seiten, 38 DM.
ISBN 3-411-03116-6

Für den Anfänger in Sachen Programmierung gedacht, basiert dieses Buch auf den Erfahrungen, die die Autoren während der Erstellung und Weiterentwicklung eines Programmierkurses an einer renommierten deutschen Fernuniversität gemacht haben. Erklärtes Ziel ist es, Schüler und Studenten unabhängig von Programmiersprache und Rechnersystem in die Programmierung einzuführen. Ergänzt durch eine Vielzahl von Blockschemata, Tabellen und Zeichnungen wird der Bereich vom Problem über den Algorithmus zu höheren Programmkonstrukten professionell und verständlich dargestellt.
Da sich die Programmierung letztendlich nur anhand einer Programmiersprache erläutern läßt, wurde für die Beispiele in diesem Buch die Sprache Pascal gewählt. Es kann, da alle wichtigen in Pascal enthaltenen Sprachmittel enthalten sind, zusätzlich als ein Lehrbuch für diese Sprache verwendet werden.
Aufgrund der klaren und leicht nachvollziehbaren Erklärungen und Beispiele eignet sich dieses Buch nicht nur für Schule, Studium und Computerseminare, sondern auch für das Selbststudium. Es ist, kurz gesagt, ein empfehlenswertes Lehrbuch, das auch gehobenen Ansprüchen voll gerecht wird.

Volker Bollig

Assemblersprache und Hardware des IBM-PC

Von Sargent, Shoemaker und Stelzer. 472 Seiten, Kart. 78 DM. Addison-Wesley Verlag, Bonn.
ISBN 3-925118-28-4

Die Autoren geben nach einem kurzen Überblick über die Hardware des IBM-PCs eine Einführung in den 8086-Assembler. Über 150 Seiten werden diesem Thema gewidmet. Es ist erfreulich, daß nicht nur wie in vielen anderen Veröffentlichungen über den 8088/8086-Assembler eine kurze Erklärung der einzelnen Befehle erfolgt, sondern immer gut kommentierte Programm-Module gezeigt werden. Themen für fortgeschrittene Assemblerprogrammierer wie Multisegment-Programme, modulare Programmierung, bedingte Assemblierung, Makros und Schnittstellen zwischen höheren Sprachen und dem Assembler kommen ebenfalls nicht zu kurz. Wie neue Ein-/Ausgabetreiber in DOS eingebunden werden, zeigen die Autoren am Beispiel eines Druckertreibers für den seriellen Ausgabekanal.
Bevor die wichtigsten Bausteine des PCs beschrieben werden, erfolgt eine kurze Einführung in die Digitalelektronik, die tatsächlich bei Adam und Eva – der Diode und dem Transistor – beginnt. Zum Thema Messen und Regeln geben die Autoren Informationen über A/D- und D/A-Umsetzer, Schrittmotoren, Signalerzeugung und -erkennung. Zum Schluß weist eine umfangreiche Übersicht auf weiterführende Literatur hin. Eine Diskette mit allen in diesem Buch aufgelisteten Programmen ist erhältlich. St

Praxis der DFÜ

Alex Pütz, „Praxis der Datenfernübertragung". 232 Seiten, 79 Abbildungen und 25 Tabellen. Lwstr-kart. Franzis-Verlag, München. 48 DM.
ISBN 3-7723-8391-2

„Wegweiser durch die Datennetze, Mailboxen und Bestimmungen" ist der mit einem ausgeprägten Sinn für Understatement gewählte Untertitel dieses Buches, läßt es doch den zukünftigen DFÜler als zuverlässiger Führer von A wie Adapter bis Z wie ZZF in keiner Phase der Errichtung einer DFÜ-Station allein.
Angefangen bei den Grundlagen beschreibt der Autor Schritt für Schritt den Aufbau einer Datenfernübertragungseinrichtung auf der Basis eines C-64 und beweist, daß die DFÜ kein exklusives Hobby von Besitzern teurer Computer sein muß.
Die Bestimmungen der Deutschen Bundespost werden genauso ausführlich behandelt wie die verschiedenen Datenkommunikationsnetze Telefon, Telex und Datex.
Breiten Raum in diesem Buch nimmt der Abschnitt über die Hardware ein. Der Autor erklärt nicht nur die Funktionsweise von Modems und Akustikkopplern, sondern stellt auch eine repräsentative Auswahl mit technischen Daten und Preisen vor.
Mit leicht verständlichen Listings in Basic werden Terminal-, Umwandlungs- und Dienstprogramme erläutert.
Das Buch wird durch eine Auflistung interessanter Mailboxen, Datenbanken sowie Schaltplänen und allen notwendigen Tabellen ergänzt.
Alles in allem ein Buch, das seinem Anspruch, ein Wegweiser zu sein, mehr als gerecht wird. *Volker Bollig*

Wordstar für PCs und Kompatible

Von Günther Jürgensmeier. 542 Seiten, über 600 Abbildungen. Kart. 59 DM. Verlag Markt & Technik AG, Haar bei München.
ISBN 3-89090-224-3

Wordstar, schon seit einigen Jahren auf dem Markt, gehört immer noch zu den beliebtesten Textverarbeitungsprogrammen. Nicht zuletzt deshalb, weil es heute zu einem günstigen Preis für Heim- und Personal Computer angeboten wird. Wordstar erfordert allerdings vom Benutzer eine intensive Einarbeitung, wenn man dieses Programm in seiner ganzen Leistungsfähigkeit nutzen möchte. Und hier soll das Buch die nötige Hilfestellung geben. Der Autor geht ausführlich auf alle Wordstar-Befehle ein, zahlreiche Abbildungen erleichtern die Verständlichkeit noch zusätzlich.
Erläutert wird auch, wie man Daten und Texte aus anderen Standardprogrammen übernehmen kann, außerdem, wie Wordstar an den PC und die individuellen Peripheriegeräte angepaßt wird. Und nicht zuletzt ist dem Installationsprogramm ein breiter Platz eingeräumt, so daß der Benutzer sich „seinen" Wordstar optimal einrichten kann. Das Buch ist für alle Wordstar-Versionen vom Junior-Wordstar bis zum Wordstar 3.4 geeignet und behandelt auch den Einsatz von Mailmerge. In der Tat ein nahezu unentbehrliches Handbuch für jeden ernsthaften Anwender. *Fle.*

Stefan Hammes

Prolog

Eine Einführung

Hand in Hand mit der künstlichen Intelligenz befindet sich die Programmiersprache Prolog auf dem Vormarsch – und macht auch vor der Welt der Mikrocomputer nicht halt. Wohl nicht ganz zu Unrecht, eröffnet das „unbekannte Wesen" doch neue Möglichkeiten und Lösungsansätze. Einen Einblick in das Prinzip, das diesem „Exoten" zugrunde liegt – in seiner Sprachstruktur hat Prolog kaum etwas mit den herkömmlichen Programmiersprachen gemeinsam –, gibt dieser Beitrag.

Die Sprache Prolog erfreut sich seit ihrer Entstehung im Jahre 1970 reger Beliebtheit, wenngleich sie in der Anfangszeit fast ausschließlich auf größeren Rechenanlagen Anwendung fand. Erst in neuerer Zeit sind Implementationen auf kleineren Systemen verfügbar (z. B. CP/M, C-64).
Aus diesem Grunde vegetierte Prolog in der Mikrocomputerwelt lange vor sich hin – dies aber völlig zu Unrecht, wie wir noch sehen werden. Prolog ist aufgrund seiner Struktur besonders für folgende Aufgabenbereiche geeignet, bei denen konventionelle Sprachen oft klein beigeben müssen:

– Symbolische Mathematiksysteme
– Expertensysteme
– Übersetzung natürlicher Sprachen
– Strukturanalyse
– logisches Schließen.

Alles Problemstellungen der künstlichen Intelligenz (AI), wie man sieht. Da Prolog offensichtlich für Aufgaben in der AI entworfen wurde, weicht die Sprachstruktur erheblich von Sprachen wie Pascal, C u. a. ab. Jemand, der an diese Sprachen gewöhnt ist, wird Prolog am Anfang sicherlich als eine große Umstellung empfinden. Für einen ungeübten Programmierer dürfte der Einstieg einfacher sein, da er in der Regel die Einschränkungen gar nicht kennt, die konventionelle Sprachen ihm auferlegen – Einschränkungen, die in Prolog nicht bestehen.
Ein Programm in Prolog ähnelt kaum der Art und Weise, wie man einen Algorithmus in konventionellen Sprachen spezifiziert. In Prolog „fragt" der Programmierer, welche Beziehungen zwischen Objekten in seinem Problem vorkommen und welche davon wahr sind. Prolog ist also eine beschreibende Sprache und gibt nicht unbedingt die Schritte vor, die zur Lösung eines Problems nötig sind. Wenn eine Berechnung ausgeführt wird, so kommt diese zum einen durch die deklarative Semantik von Prolog, zum anderen durch neue Fakten, die aus vorhandenen Objekten geschlossen werden können und nur teilweise durch die explizite Steuerungsinformation von seiten des Programmierers zustande.

Terme

In Prolog gibt es nur eine einzige Datenstruktur, den Term, aus dem alle Daten, ja sogar Prolog-Programme selbst, konstruiert werden können. Ein Prolog-Programm besteht aus einzelnen Sätzen, wobei jeder Satz entweder eine Tatsache über ein bestimmtes Objekt beinhaltet oder eine Vorschrift angibt, wie die Lösung eines Problems zu den vorhandenen Objekten in Beziehung steht bzw. aus ihnen abgeleitet werden kann.
Eine Beziehung kann etwa folgendermaßen aussehen: „Otto besitzt einen Computer". Die Beziehung 'besitzen' besteht also zwischen dem Objekt 'Otto' und dem Objekt 'Computer'. Auch die Reihenfolge in der Beziehung ist wichtig, denn der Computer besitzt Otto ja nicht.
Ein Beispiel für eine Frage: „Besitzt Otto den Computer ?". Hier erwarten wir 'ja' oder 'nein' als Antwort.
Ein Faktum ist eine Aussage über Eigenschaften von Objekten oder Beziehungen zwischen ihnen. Eine Vorschrift (oder Regel) lautet beispielsweise: „Zwei Menschen sind Schwestern, wenn sie beide weiblich sind und die gleichen Eltern haben". Sie sagt aus, was „Schwester" bedeutet und gibt außerdem an, wie man herausfinden kann, ob zwei Personen Schwestern sind.
Zusammenfassend sieht das Programmieren in Prolog also so aus:

1. Deklarieren von Fakten über Objekte und ihre Beziehungen.
2. Definieren von Regeln über Objekte und ihre Beziehungen.
3. Fragen stellen über Objekte und ihre Beziehungen.

Prolog ist eine Dialogsprache (ähnlich LISP), d. h. man gibt etwas ein und Prolog zeigt eine Reaktion; dann beginnt der Kreislauf von vorne.

Konstanten

Konstanten dienen dazu, bestimmte Objekte oder Beziehungen zu benennen. Es gibt zwei Arten von Konstanten: Atome und ganze Zahlen. Bei den Atomen lassen sich wiederum zwei Sorten unterscheiden, nämlich solche aus Buchstaben und Ziffern und solche aus Sonderzeichen. Die erstgenannten müssen mit einem kleinen Buchstaben beginnen und können ein Unterstreichungszeichen enthalten. Letztere hingegen bestehen nur aus Sonderzeichen. Atome in einfachen Anführungszeichen können alle Zeichen enthalten. Beispiele:
otto, =, ->, 'Hallo!', a12.

Zahlen – die zweite Art von Atomen – sind endlich alte Bekannte aus normalen Sprachen, die Grundlage für arithmetische Operationen. Oft ist ihr Wertebereich auf 16 Bit beschränkt, jedoch wird Prolog vorzugsweise sowieso auf Gebieten angewandt, wo einfache Ganzzahlen-Arithmetik ausreicht.

Fakten

Um die Beziehung: „Otto besitzt einen Computer" in Prolog auszudrücken, muß man wie folgt formalisieren:
besitzt(otto, computer).

– Da die Namen der Objekte und Beziehungen Konstanten sind, müssen sie mit einem Kleinbuchstaben beginnen.
– Die Beziehung steht an erster Stelle, dahinter die Objekte. Beide werden durch Kommata voneinander abgegrenzt und in ein Klammerpaar eingeschlossen.
– '.' beendet das Faktum.

Ein weiteres Beispiel: schnell(computer).
In Deutsch: „Der Computer ist schnell".
Der Name der Beziehung heißt Prädikat,

die Namen in der Klammer sind die Argumente. Im Klartext: 'besitzt' ist ein Prädikat mit zwei Argumenten, 'schnell' ist eines mit einem Argument. Mehrere Fakten zusammen bilden eine Datenbasis (später werden auch Regeln dazugehören).

Formulierung von Fragen

Wenn in der Datenbasis ein paar Fakten sind, kann man über sie Fragen stellen. Dafür gibt es ein spezielles Atom: ?- Hierauf antwortet Prolog mit ja oder nein. Mit den Fakten, die nun schon im System sind, könnte man z. B. folgende Frage stellen:
?- besitzt(otto, computer).
In Deutsch: „Besitzt Otto einen Computer ?".
Nachdem man die Frage eingetippt hat, sucht Prolog in der Datenbasis die Fakten, die zum Faktum in der Frage passen. Zwei Fakten passen zueinander, wenn ihre Prädikate gleich sind und jedes ihrer Argumente gleich ist. Wenn ein passendes Faktum gefunden wurde, antwortet Prolog mit 'ja', ansonsten mit 'nein'. Auf unsere Frage würde die Antwort also 'ja' lauten. Bei
?- besitzt(karl, computer).
wäre sie 'nein'.

Wenn man wissen möchte, wer alles einen Computer besitzt, so könnte man fragen: „Besitzt Otto einen Computer ?", „Besitzt Karl einen Computer ?" usw., und Prolog würde immer mit 'ja' oder 'nein' antworten. Viel einfacher hingegen wäre es, zu fragen: „Wer besitzt einen Computer ?". Auch das ist möglich, denn dafür gibt es in Prolog Variablen.

Variablen

Diese Art von Termen sieht den Konstanten auf den ersten Blick sehr ähnlich, mit dem kleinen Unterschied, daß sie mit einem Großbuchstaben oder einem Unterstreichungszeichen beginnen müssen. Eine mögliche Frage wäre: „Besitzt X einen Computer ?". Zu diesem Zeitpunkt weiß man noch nicht, wofür X hier steht.
Variablen können in Prolog zwei Zustände annehmen:

1. Sie können frei sein. In diesem Fall ist das, wofür die Variable steht, noch nicht bekannt.
2. Sie sind gebunden, d. h. die Variable steht für ein bestimmtes Objekt.

Geben wir aber zuerst noch ein Faktum ein:
besitzt(karl, computer).
Wenn man nun fragt: besitzt(X, computer), dann ist X zunächst eine freie Variable, da sie noch für kein Objekt steht. Daraufhin sucht Prolog in der Datenbasis ein Faktum, welches auf die Frage paßt. Hierbei paßt X (da noch frei) auf jedes Argument eines Faktums in der Datenbasis in derselben Position. Konkret: Prolog sucht ein Faktum, das als Prädikat 'besitzt' hat, und als zweites Argument 'computer'. Das erste Argument kann etwas ganz Beliebiges sein. Wenn solch ein Faktum gefunden wurde, steht X nun für das erste Argument und ist somit gebunden. Da Prolog die Datenbasis in der Reihenfolge durchgeht, in der sie eingegeben worden ist, trifft es zunächst auf:
besitzt(otto, computer).
X steht nun für 'otto'; man sagt auch X ist an 'otto' gebunden. Die Position, in der das passende Faktum gefunden wurde, wird in der Datenbasis markiert.
Beim Auffinden solch eines passenden Faktums druckt Prolog die Objekte, für die die Variable(n) stehen, aus, in unserem Falle also:
X=otto
und wartet auf weitere Eingaben. Wenn man nun Return tippt, stellt Prolog die Suche ein, tippt man jedoch ein Semikolon ein, geht sie weiter, und zwar von der Position an, die vorher markiert wurde, also nicht vom Beginn der Datenbasis aus. Wenn wir in unserem Beispiel also ';' gedrückt hätten, würde Prolog die Variable X wieder auf 'frei' setzen und die Suche wie oben beschrieben fortsetzen.
Das nächste Faktum war: besitzt(karl, computer). X steht nun für 'karl' und es wird X=karl ausgegeben. Dabei wurde wieder die Position des Faktums markiert, damit beim nächsten ';' die Suche von diesem Punkt aus gestartet werden kann. Wenn wir also nochmal ';' eingeben, geht die Suche weiter, aber es wird nichts mehr gefunden, daher druckt Prolog 'nein' aus.

Verknüpfung von Fragen

Zur Erklärung der Konjunktion führen wir zuerst ein paar Fakten ein:
mögen(otto, bier).
mögen(otto, wein).
mögen(susi, wein).
mögen(susi, otto).

Man könnte jetzt fragen, ob sich 'otto' und 'susi' gegenseitig mögen:
?- mögen(otto, susi), mögen(susi, otto).
Das Komma ist die Konjunktion und wird 'und' ausgesprochen. Es wird zur Verbindung einzelner Fragen benutzt, die alle mit 'ja' beantwortet werden müssen, damit die Gesamtfrage mit 'ja' beantwortet werden kann. In unserem Falle lautet die Antwort 'nein', da zwar 'susi' 'otto' mag, aber nicht umgekehrt. Wenn man z. B. wissen möchte, was 'otto' und 'susi' beide mögen, so wird man wieder auf die Variablen zurückgreifen.

Die Formulierung in Prolog:
?- mögen(otto, X), mögen(susi, X).
D. h.: Suche etwas, was 'otto' mag, dann schaue nach, ob 'susi' das auch mag.
Prolog beginnt mit der ersten Teilfrage. Wenn ein passendes Faktum gefunden wurde, wird X an dessen zweites Argument gebunden und ist folglich nicht mehr frei. Gleichzeitig wird die Position in der Datenbasis markiert. In unserem Falle steht X für 'bier'. X behält diesen Wert für den Rest der Gesamtfrage. Die zweite Teilfrage heißt also nun:
mögen(susi, bier).
Dieses Faktum existiert aber nicht , daher wird die zweite Teilfrage mit 'nein' beantwortet, X wird wieder frei und die Suche wird hinter der markierten Position fortgesetzt. Das nächste passende Faktum ist:
mögen(otto, wein).
X steht jetzt für 'wein' und die Position wird wieder markiert. Die zweite Teilfrage lautet entsprechend:
mögen(susi, wein).
Die Suche beginnt von neuem am Anfang der Datenbasis, da vorher keine Position markiert wurde – und ist erfolgreich. Wiederum wird die Position markiert. Da X für 'wein' steht und beide Teilfragen mit 'ja' beantwortet wurden, druckt Prolog
X=wein
und wartet auf weitere Befehle. Die beiden Markierungen stehen bei:
mögen(otto, wein). und
mögen(susi, wein).
Offensichtlich besitzt jede Teilfrage ihre eigene Markierung. Wenn wir wissen wollen, ob 'otto' und 'susi' noch mehr gemeinsam mögen, tippen wir einfach ';' ein, und die Suche wird nach dem gleichen Schema fortgesetzt – und zwar ab der Position, die die jeweiligen Markierungen bestimmen.

Zusammenfassend läßt sich also festhalten: Eine Konjunktion von zwei oder mehr Teilfragen wird von Prolog von links nach rechts bearbeitet. Wenn eine Teilfrage bejaht wird, setzt Prolog eine entsprechende Markierung in die Datenbasis und eventuell freie Variablen werden gebunden. Auch in weiteren Teilfragen bleiben diese gebunden. Anschlie-

ßend versucht Prolog die (nach rechts) folgende Teilfrage zu beantworten. Dabei beginnt die Suche am Anfang der Datenbasis. Bei der Bejahung einer Teilfrage wird eine korrespondierende Markierung in die Datenbasis gesetzt, für den Fall, daß die Teilfrage später nochmal benötigt wird.
Wird eine Teilfrage verneint, geht Prolog zum linken Nachbarn zurück und startet die Suche bei den jeweiligen Markierungen, um herauszufinden, ob eventuell noch irgendwelche anderen Fakten passen. Zudem müssen die in einer Teilfrage gebundenen Variablen wieder frei werden, damit sie auf beliebige Argumente passen. Wenn die ganz links stehende Teilfrage verneint wird, so gilt die ganze Konjunktion als verneint.
Dieses Prinzip – der Versuch zur Beantwortung von Fragen und ein wiederholtes Zurückgehen bei Mißerfolg – nennt man Backtracking, eines der zentralen Konzepte von Prolog.

Beantwortung durch Regeln

Angenommen, man wolle ausdrücken, daß 'otto' alle Computer besitzt. Dies kann man so formulieren:

```
besitzt(otto, apple).
besitzt(otto, mc68000).
usw.
```

Das ist ziemlich viel Schreibarbeit und außerdem meist nicht vollständig. Besser ist: „Otto besitzt jedes Objekt, daß ein Computer ist". Das ist eine Regel über das, was 'otto' besitzt. In Pseudo-Prolog würde das wie folgt lauten: „X wird von Otto besessen, wenn X ein Computer ist".
In richtigem Prolog heißt es:

```
besitzt(otto, X) :- computer(X).
```

:- ist wieder ein Atom, das eine Funktion auszuführen hat. Der Regelkopf, hier 'besitzt(otto, X)', beschreibt das, was die Regel definiert, der Regelkörper hingegen, hier 'computer(X)', beschreibt die Konjunktion von Teilfragen (in unserem Beispiel nur eine Teilfrage), die bejaht werden müssen, damit der Regelkopf wahr wird. Damit diese Regel anwendbar wird, müssen noch ein paar Fakten eingegeben werden, z. B.:

```
computer(apple).
computer(mc-68000).
computer(c64).
```

Wenn wir jetzt eingeben:

```
?- besitzt(otto, X).
```

dann wird der Reihe nach ausgegeben:

```
X=apple
X=mc68000
X=c64
no
```

Nach jeder Antwort ist ';' zu tippen.
Zur Erklärung dient ein Beispiel, bei dem es um Verwandschaftsverhältnisse geht. Zunächst ein paar Fakten:

```
mann(otto).
mann(karl).
frau(susi).
frau(karin).
eltern(karl, karin, otto).
eltern(susi, karin, otto).
```

eltern (x,y,z) soll heißen: die Eltern von x sind y und z. Nun eine Regel: schwester(X, Y) soll heißen, daß X Schwester von Y ist. Dies gilt offensichtlich, wenn X weiblich ist, X die Mutter M und den Vater V hat und Y die gleiche Mutter und den gleichen Vater wie X hat. In Prolog:

```
schwester(X, Y) :- frau(X),
                   eltern(X, M, V),
                   eltern(Y, M, V).
```

Eine mögliche Frage wäre nun:

```
?- schwester(susi, karl).
```

X wird an 'susi', Y an 'karl' gebunden. Dann wird die erste Teilfrage betrachtet, die da lautet: frau(susi). Dies ist wahr, daher wird zur nächsten Teilfrage übergegangen. Sie lautet: eltern(susi, M, V). M und V sind hier frei, daher passen sie auf beliebige Argumente. Ein passendes Faktum ist eltern(susi, karin, otto). M steht jetzt für 'karin', V für 'otto'. Da auch diese Teilfrage zu bejahen ist, wird mit der nächsten und letzten Teilfrage weitergemacht. Mit den jetzt gebundenen Variablen heißt sie: eltern(karl, karin, otto). Dies paßt auf unser fünftes Faktum, somit antwortet Prolog mit 'ja'.
Wenn man – jetzt schon etwas komplizierter – wissen will, wessen Schwester 'susi' ist, fragt man:

```
?- schwester(susi, X).
```

X in der Regel wird an 'susi' gebunden, Y bleibt frei, da X in der Frage frei ist. Die erste Teilfrage: frau(susi) wird bejaht; daher geht es mit 'eltern(susi, M, V)' weiter. Das paßt wie vorhin auf eltern(susi, karin, otto). M und V sind nun gebunden. Da auch die zweite Teilfrage bejaht wurde, lautet der dritte Schritt eltern(Y, karin, otto). Y ist hier noch frei, da das zweite Argument bei der Fragestellung auch frei war. Diese Teilfrage paßt auf eltern(karl, karin, otto) und Y wird an 'karl' gebunden. Da auch diese letzte Teilfrage bejaht wurde, ist die Konjunktion wahr. X in der Frage ist äquivalent zu Y in der Regel, also wird X auch an 'karl' gebunden und Prolog antwortet: X=karl

Arithmetik

Als arithmetische Operatoren stehen +, -, *, / und mod zur Verfügung, für Vergleiche =, \=, >, <, >= und =<. Als Beispiel möge hier die Bevölkerungsdichte in einigen Ländern dienen. Zunächst wieder ein paar Fakten, die aussagen, wieviele Millionen Einwohner ein Land hat:

```
bev(usa, 203).
bev(indien, 548).
bev(china, 800).
```

Nun die Fläche dieser Länder in Millionen Quadratkilometer:

```
flaeche(usa, 6).
flaeche(indien, 2).
flaeche(china, 4).
```

Um die Bevölkerungsdichte zu berechnen, muß man die Bevölkerungszahl durch die Fläche teilen. Das leistet die folgende Regel:

```
dichte(X, Y) :- bev(X, B),
                flaeche(X, F),
                Y is B/F.
```

Das kann man so lesen: Die Bevölkerungsdichte von Land X ist Y, wenn die Bevölkerung von X B ist und die Fläche von X F ist und Y errechnet wird durch B/F. Der Operator 'is' weist einer freien Variablen links das Resultat eines mathematischen Ausdrucks rechts zu. Nun lautet beispielsweise die Antwort des Rechners auf die Frage

```
?- dichte(china, X).
X=200.
```

200 bedeutet hier Menschen pro Quadratkilometer. Die Frage

```
?- dichte(brd, X).
```

ergibt 'no', da hierzu kein Eintrag in der Datenbasis zu finden ist.

Strukturen

Eine Struktur als eine weitere Sorte von Termen ist ein einzelnes Objekt, das aus mehreren anderen Objekten besteht. Im weitesten Sinne ist dies mit einem Record in Pascal vergleichbar. Man kann schreiben: besitzt(otto, mc68000).
Um etwas Genaueres über den Computer auszusagen, wird dies abgewandelt zu: besitzt(otto, computer(mc68000, motorola)).
Und noch präziser: besitzt(otto, computer(mc68000, firma(motorola, usa))).
Nehmen wir an, wir hätten viele solcher Strukturen eingegeben und wollten nun wissen, welche von Ottos Computern aus USA kommen. Um es herauszufinden, wird die Frage:

```
?- besitzt(otto, computer(X, firma(Y, usa))).
```

eingegeben.
X gibt in der Antwort den Computertyp, Y den Firmennamen wieder. Wer sich

Prof. Dr. Jochen Ziegenbalg

Die gläserne Gleitkommazahl

Ein Programm zur Darstellung des Bitmusters von Gleitkommazahlen (in Turbo-Pascal)

Obwohl das Wort „Computer" nichts anderes als „Rechner" bedeutet, weisen die meisten Computer gerade im Bereiche des Rechnens große Schwachstellen auf. Dies ist letztlich in der Tatsache begründet, daß sich die Zahlen nur in sehr unzulänglicher Weise im Computer darstellen lassen. Denn es gibt unendlich viele Zahlen; der Computer kann aber nur endlich viele verschiedene Zustände einnehmen.

Im Bereich der reellen Zahlen gibt es besonders „widerborstige" Exemplare mit nichtabbrechender und nicht periodischer Dezimalbruchentwicklung, etwa:

PI = 3,141592653589793...

Allein die Darstellung dieser Zahl als Dezimalzahl würde den Speicher jedes Computers sprengen. Deshalb werden die Dezimalzahlen im Speicher des Computers in einem speziellen, ökonomischen Format als „Gleitkommazahlen" abgelegt. Diese Methode führt aber zu unvermeidbaren Rundungsfehlern, und zwar:

☐ bei der Eingabe von Dezimalzahlen und der damit verbundenen Umwandlung in das Gleitkommaformat;

☐ beim internen Rechnen mit Gleitkommazahlen.

In der Gleitkommadarstellung werden die Zahlen (ähnlich wie in der sogenannten wissenschaftlichen Notation) durch Exponent und Mantisse beschrieben. Allerdings werden Exponent und Mantisse im Prinzip im Binärsystem (Zweiersystem) dargestellt. Außerdem wird für Exponent und Mantisse von vorn herein und unabhängig von der wirklichen Größe der Zahl jeweils ein Feld fester Länge reserviert; in Turbo-Pascal ein Byte für den Exponenten und fünf Bytes für die Mantisse.

Diese Gleitkommazahlen, ihre interne Speicherung und die damit verbundenen Eingabe- und Rechenfehler in der Gleitkommaarithmetik sind in Pascal (wie in anderen Programmiersprachen auch) ein besonders undurchsichtiges, meist extrem schlecht dokumentiertes Kapitel.

Gleitkommazahlen durch ein Programm analysieren

Das im folgenden abgedruckte Programm dient dem Zweck, die Darstellung der Gleitkommazahlen in Turbo-Pascal transparent zu machen. Mit seiner Hilfe kann sich der Benutzer das intern verwendete Bitmuster von Gleitkommazahlen erklären und an selbst gewählten Beispielen aufzeigen lassen. In der vorliegenden Form ist das Programm unter CP/M (80 und 3.0) gelaufen; es dürfte ohne Änderung auch unter MS-DOS lauffähig sein.

Nach einer allgemeinen Erläuterung der internen Struktur von Gleitkommazahlen durch das Programm kann der Benutzer eine beliebige (Gleitkomma-)Zahl eingeben. Von dieser Zahl werden Exponent, Mantisse (in normalisierter Form) und die korrigierte Mantisse ausgedruckt. Schließlich werden die als

```
 1:  Program Bitmuster_von_Gleitkommazahlen;

 3:  var antwort: char;
 4:      ausdatei: Text;
 5:      ausgeraet: string[10] ;

 7:  procedure allgemeine_erlaeuterung;

 9:  procedure weiter;
10:  var w: char;
11:  begin
12:    writeln(ausdatei, '');
13:    if ausgeraet = 'Con:' then
14:         begin
15:           write('Weiter mit <RETURN>: ');
16:           readln(w);
17:         end;
18:    clrscr;
19:  end;

21:  begin  (* allgemeine_erlaeuterung *)
22:    clrscr;
23:    writeln(ausdatei, 'Erlaeuterung des Bitmusters von Gleitkommazahlen ');
24:    writeln(ausdatei, '(im 6 Byte Format von Turbo Pascal)');
25:    writeln(ausdatei, '');
26:    writeln(ausdatei, 'Schematische Darstellung der 6 Bytes: ');
27:    writeln(ausdatei, '');
28:    writeln(ausdatei, '1. Byte    2. Byte    3. Byte    ' ,
29:                      '4. Byte    5. Byte    6. Byte');
30:    writeln(ausdatei, '');
31:    writeln(ausdatei, 'ABCD EFGH  IJKL MNOP  QRST UVWX  ' ,
32:                      'abcd efgh  ijkl mnop  qrst uvwx');
33:    writeln(ausdatei, '');
34:    writeln(ausdatei, 'Im ersten Byte  ABCD EFGH  steht der um die' ,
35:                      ' Zahl 128 erhoehte');
36:    writeln(ausdatei, 'Zweier-Exponent.');
37:    writeln(ausdatei, '(Im Computer-Jargon: "Die Darstellung des' ,
38:                      ' Exponenten erfolgt mit');
39:    writeln(ausdatei, 'einem $80 offset" - vgl. Turbo Pascal ' ,
40:                      'Handbuch, Ziffer 22.18.1.2.)' );
41:    writeln(ausdatei, '');
42:    writeln(ausdatei, 'Mit anderen Worten:  das am weitesten links' ,
```

Bild 1. Das Programm, das die Gleitkomma-Zahlen untersucht

```
43:                         ' stehende Bit ');
44:     writeln(ausdatei, '(d.h. das Bit A in der obigen Darstellung)' ,
45:                         ' kann folgender-');
46:     writeln(ausdatei, 'massen als das Vorzeichen des Exponenten' ,
47:                         ' angesehen werden: ');
48:     writeln(ausdatei, '');
49:     writeln(ausdatei, '       Bit A = 1:   der Exponent ist nichtnegativ ');
50:     writeln(ausdatei, '       Bit A = 0:   der Exponent ist negativ ');
51:     weiter;
52:     writeln(ausdatei, 'Die Mantisse ist in Byte Nr. 2 bis Byte Nr. 6' ,
53:                         ' gespeichert; ');
54:     writeln(ausdatei, 'und zwar in der Reihenfolge: ');
55:     writeln(ausdatei, 'Least Significant Byte (LSB) ---> Most' ,
56:                         ' Significant Byte (MSB)');
57:     writeln(ausdatei, '');
58:     writeln(ausdatei, 'Dem Stellenwert nach aufgeschrieben lautet' ,
59:                         ' die Mantisse also ');
60:     writeln(ausdatei, 'folgendermassen: ');
61:     writeln(ausdatei, '');
62:     writeln(ausdatei, 'qrst uvwx  ijkl mnop  abcd efgh  QRST UVWX  IJKL MNOP');
63:     writeln(ausdatei, '');
64:     writeln(ausdatei, 'Die Darstellung der Mantisse ist "normalisiert".' ,
65:                         '  Das heisst, das am');
66:     writeln(ausdatei, 'weitesten links stehende Bit q der Mantisse' ,
67:                         ' wird stets als 1 gewer-');
68:     writeln(ausdatei, 'tet; unabhaengig davon, ob es tatsaechlich' ,
69:                         ' gleich 1 ist oder nicht.');
70:     writeln(ausdatei, 'Dieses  - so fuer eine andere Interpretation' ,
71:                         ' frei gewordene - ');
72:     writeln(ausdatei, 'Bit wird nun als das Vorzeichenbit der Mantisse' ,
73:                         ' (und somit der');
74:     writeln(ausdatei, 'gesamten Zahl) verwendet.  Dabei bedeutet: ');
75:     writeln(ausdatei, '');
76:     writeln(ausdatei, '       Bit q = 0:   die Zahl ist nichtnegativ');
77:     writeln(ausdatei, '       Bit q = 1:   die Zahl ist negativ');
78:     writeln(ausdatei, '');
79:     writeln(ausdatei, 'Die Mantisse lautet in Wirklichkeit also stets' ,
80:                         ' folgendermassen: ');
81:     writeln(ausdatei, '');
82:     writeln(ausdatei, '1rst uvwx  ijkl mnop  abcd efgh  QRST UVWX  IJKL MNOP');
83:     weiter;
84:   end;  (* allgemeine_erlaeuterung *)

86:   procedure beispiel;
87:   type bitstring = string[53];
88:        real_maske  =  record
89:                         case boolean of
90:                           true:  (r: real);
91:                           false: (b: array[0..5] of byte);
92:                       end;

94:   var  x: real_maske;
95:        r, vorzeichen: real;
96:        i, exp: integer;
97:        s, m: bitstring;

99:   function int_wert(s: bitstring; l: integer): integer;
100:    var j, stelle, summe: integer;
101:    begin
102:      stelle := 1;
103:      summe := 0;
104:      for j := l downto 1 do
105:          begin
106:            if s[j] = '1' then summe := summe + stelle;
107:            stelle := 2 * stelle;
108:          end;
109:      int_wert := summe;
110:    end;

112:  function real_wert(s: bitstring; l: integer): real;
113:  var j: integer;
114:        stelle, summe: real;
115:    begin
116:      stelle := 1;
117:      summe := 0;
118:      for j := l downto 1 do
119:          begin
120:            if s[j] = '1' then summe := summe + stelle;
121:            stelle := 2 * stelle;
122:          end;
123:      real_wert := summe;
124:    end;
126:  function gesamtwert(r: real; k: integer): real;
127:    var j: integer;
128:    begin
129:      if k < 40 then for j := 1 to (40 - k) do  r := r / 2 ;
130:      if k > 40 then for j := 1 to (k - 40) do  r := 2 * r ;
131:      gesamtwert := r;
132:    end;

134:  function bitmuster_des_reals(r: real_maske): bitstring;
135:    type bitwort = string[8];
136:    var b: byte;

138:    function bitmuster_des_bytes(b: byte): bitwort;
139:      var i: integer;
140:          s: bitwort;
141:          s1: string[1];
142:          b1: byte;
143:      begin
144:        b1 := b;
145:        s := '' ;
146:        for i := 1 to 8 do
147:          begin
148:            str( ( b1 and 1 ) , s1 );
149:            s :=  concat(s1, s);
150:            b1 := b1 shr 1 ;
151:          end;
152:        bitmuster_des_bytes := s;
153:      end;

155:  begin  (* bitmuster_des_reals *)
156:    bitmuster_des_reals :=
157:       concat(bitmuster_des_bytes( r.b[0] ) , ' ' ,
158:              bitmuster_des_bytes( r.b[1] ) , ' ' ,
159:              bitmuster_des_bytes( r.b[2] ) , ' ' ,
160:              bitmuster_des_bytes( r.b[3] ) , ' ' ,
161:              bitmuster_des_bytes( r.b[4] ) , ' ' ,
162:              bitmuster_des_bytes( r.b[5] ) );
163:  end;  (* bitmuster_des_reals *)

165:  function mantisse(s: bitstring): bitstring;
166:    var j: integer;
167:        m: bitstring;
168:    begin
169:      m := '' ;
170:      for j := 5 downto 1 do m := concat(m, copy(s, 9*j + 1, 8));
171:      if copy(m, 1, 1) = '1' then vorzeichen := -1 else vorzeichen := 1;
```

```
172:       mantisse := concat( '1', copy(m, 2, 39));
173:     end;

175:   begin  (* beispiel *)
176:     write('Ein Beispiel:   x = ');  readln(x.r);
177:     if not (ausgeraet = 'Con:') then
178:        begin
179:          writeln(ausdatei, 'Ein Beispiel:       x = ' , x.r );
180:          writeln(ausdatei, 'Festkomma-Darstellung: ' , x.r : 18 : 6);
181:        end;
182:     writeln(ausdatei, '');
183:     s := bitmuster_des_reals(x);
184:     m := mantisse(s);
185:     writeln(ausdatei, 'Bitmuster (Reihenfolge wie im Speicher): ');
186:     writeln(ausdatei, '');
187:     writeln(ausdatei, 'Exponent LSB                                    MSB');
188:     writeln(ausdatei, s);
189:     if copy(s, 1, 8) = '00000000' then
190:        begin
191:          writeln(ausdatei, '');
192:          writeln(ausdatei, 'Die Null ist dadurch gekennzeichnet, dass');
193:          writeln(ausdatei, 'das Exponenten-Byte gleich 00000000 ist.');
194:          exit;
195:        end;
196:     writeln(ausdatei, '^                                              ^');
197:     writeln(ausdatei, '!                                              !');
198:     writeln(ausdatei, 'mit $80-offset              Vorzeichen:  0 - nichtnegativ');
199:     writeln(ausdatei, '                                         1 - negativ');
200:     exp := int_wert(copy(s, 1, 8), 8) - 128 ;
201:     writeln(ausdatei, 'Wert des Zweier-Exponenten (dezimal): ' , exp);
202:     writeln(ausdatei, '');
203:     writeln(ausdatei, 'Mantisse stellengerecht: ');
204:     write(ausdatei, 'Vorzeichenbit unkorrigiert: ' );
205:     for i := 5 downto 1 do write(ausdatei, copy(s , 9*i , 9));
206:     writeln(ausdatei, '');
207:     write(ausdatei, 'Vorzeichenbit korrigiert:   ');
208:     for i := 0 to 4 do write(ausdatei, ' ' , copy(m , 8*i +1 , 8));
209:     writeln(ausdatei, ''); writeln(ausdatei, '');
210:     r := vorzeichen * real_wert(copy(m, 1, 40) , 40) ;
211:     writeln(ausdatei, 'Wert der Mantisse (dezimal): ' , r : 15 );
212:     writeln(ausdatei, '');
213:     writeln(ausdatei, 'Rueckgewandelte Zahl: ');
214:     writeln(ausdatei, 'Exponentialdarstellung: ' , gesamtwert(r, exp) );
215:     writeln(ausdatei, 'Festkomma-Darstellung:  ' , gesamtwert(r, exp):18:6);
216:   end;  (* beispiel *)

218:   begin  (* Hauptprogramm *)
219:     clrscr;
220:     writeln('Dokumentation des Bitmusters von Gleitkommazahlen');
221:     writeln;
222:     ausgeraet := 'Con:';
223:     writeln('Ausgabe auf: ');
224:     writeln;
225:     writeln('    1 = Con: (Console; Bildschirm) ' );
226:     writeln('    2 = Lst: (list device; Drucker) ');
227:     writeln;
228:     write('Bitte waehlen Sie: ');  readln(antwort);
229:     if antwort = '2' then ausgeraet := 'Lst:';
230:     assign(ausdatei, ausgeraet);
231:     rewrite(ausdatei);
232:     allgemeine_erlaeuterung;
233:     beispiel;
234:     close(ausdatei);
235:   end.
```

```
Erlaeuterung des Bitmusters von Gleitkommazahlen
(im 6 Byte Format von Turbo Pascal)

Schematische Darstellung der 6 Bytes:

1. Byte    2. Byte    3. Byte    4. Byte    5. Byte    6. Byte

ABCD EFGH  IJKL MNOP  QRST UVWX  abcd efgh  ijkl mnop  qrst uvwx

Im ersten Byte  ABCD EFGH  steht der um die Zahl 128 erhoehte
Zweier-Exponent.
(Im Computer-Jargon: "Die Darstellung des Exponenten erfolgt mit
einem $80 offset" - vgl. Turbo Pascal Handbuch, Ziffer 22.18.1.2.)

Mit anderen Worten:  das am weitesten links stehende Bit
(d.h. das Bit A in der obigen Darstellung) kann folgender-
massen als das Vorzeichen des Exponenten angesehen werden:

      Bit A = 1:  der Exponent ist nichtnegativ
      Bit A = 0:  der Exponent ist negativ

Die Mantisse ist in Byte Nr. 2 bis Byte Nr. 6 gespeichert;
und zwar in der Reihenfolge:
Least Significant Byte (LSB) ---> Most Significant Byte (MSB)

Dem Stellenwert nach aufgeschrieben lautet die Mantisse also
folgendermassen:

qrst uvwx  ijkl mnop  abcd efgh  QRST UVWX  IJKL MNOP

Die Darstellung der Mantisse ist "normalisiert".  Das heisst, das am
weitesten links stehende Bit q der Mantisse wird stets als 1 gewer-
tet; unabhaengig davon, ob es tatsaechlich gleich 1 ist oder nicht.
Dieses  - so fuer eine andere Interpretation frei gewordene -
Bit wird nun als das Vorzeichenbit der Mantisse (und somit der
gesamten Zahl) verwendet.  Dabei bedeutet:

      Bit q = 0:  die Zahl ist nichtnegativ
      Bit q = 1:  die Zahl ist negativ

Die Mantisse lautet in Wirklichkeit also stets folgendermassen:

1rst uvwx  ijkl mnop  abcd efgh  QRST UVWX  IJKL MNOP

Ein Beispiel:       x =   3.1415926530E+06
Festkomma-Darstellung:      3141592.653000

Bitmuster (Reihenfolge wie im Speicher):

Exponent LSB                                          MSB
10010110 10101100 10011100 01100010 10111111 00111111
^                                                     ^
|                                                     |
mit $80-offset                Vorzeichen:   0 - nichtnegativ
                                            1 - negativ
Wert des Zweier-Exponenten (dezimal): 22

Mantisse stellengerecht:
Vorzeichenbit unkorrigiert:  00111111 10111111 01100010 10011100 10101100
Vorzeichenbit korrigiert:    10111111 10111111 01100010 10011100 10101100

Wert der Mantisse (dezimal):  8.235496644E+11

Rueckgewandelte Zahl:
Exponentialdarstellung:    3.1415926530E+06
Festkomma-Darstellung:       3141592.653000
```

Bild 2. Eine „Analyse". Das Beispiel muß von Hand eingegeben werden

Strings dargestellten Bitmuster wieder (sozusagen als Kontrollrechnung bzw. als Plausibilitätsbetrachtung) in die interne Gleitkommadarstellung zurückverwandelt. Diese zurückverwandelte Zahl muß natürlich, abgesehen von Rundungsfehlern, mit der ursprünglichen Zahl übereinstimmen.

Die Ausgabe kann auf dem Bildschirm oder auf einem Drucker erfolgen. Ein typisches Laufbeispiel ist im Anschluß an das Programm gegeben.

Ein Trick überspielt Grenzen von Turbo-Pascal

Dieser Beitrag soll keine Einführung in Turbo-Pascal sein. Deshalb bleibt das Pascal-Programm auch im wesentlichen unerläutert. Auf einige Besonderheiten möchte ich jedoch noch eingehen:

1. „Normalerweise" kommt man nicht an die interne bitweise Darstellung von Gleitkommavariablen heran. Der folgende Trick hilft jedoch: Man schaffe eine Möglichkeit, auf die jeweilige Gleitkommazahl mit Hilfe einer zweiten Typbeschreibung (hier: array [0...5] of byte) zuzugreifen. Syntaktisch geschieht dies mit Hilfe von sogenannten „variant records". In unserem Fall ist der Typ real_maske in den Zeilen 88 bis 92 einmal als Gleitkommatyp (real) und einmal als Feld (array [0...5] of byte) beschrieben. Der Zugriff im Sinne des jeweiligen Typs erfolgt über den entsprechenden Komponentennamen: ist x eine Variable vom Typ real_maske, dann ist mit x.r ihre Gleitkommastruktur und mit x.b ihre Feldstruktur greifbar.

2. Der Datentyp „byte" ist in Turbo-Pascal ein Untertyp des Typs „integer" mit (dezimal dargestellten) Werten zwischen 0 und 255. Um an deren Bitstruktur heranzukommen, wurden die Bitoperationen „and" (logisches und) und „shr" (shift right) verwendet (Zeile 148 und 150). Auch diese Operationen sind im Turbo-Pascal-Handbuch nur sehr schlecht dokumentiert. In bezug auf shr ist (in Abschnitt 6.1.2) keine allgemeine Beschreibung sondern nur ein Beispiel gegeben. Weiterhin steht dort, daß die Operation Operanden vom Typ integer verlangt. Wie unser Programm zeigt, funktioniert sie aber offenbar auch mit Operanden vom Typ byte. (Dies ist bei Operationen, die sich auf der Bit-Ebene abspielen nicht selbstverständlich; auch wenn im Handbuch steht, daß der Datentyp byte grundsätzlich mit dem Datentyp integer verträglich ist.)

```
bit = 0...1          und
real_maske = record
                 case boolean of
                   true:  (r: real);
                   false: (b: packed array[0...47] of bit;
               end
```

3. In UCSD-Pascal hätte man das ganze Programm erheblich einfacher schreiben können. Denn mit den Typen
hätte man über den Aufruf x.b[k] unmittelbar ohne alle Umwandlungstricks auf das k-te Bite der Gleitkommadarstellung von x zugreifen können. Denn in UCSD-Pascal werden die 48 Bits in genau 6 Bytes gepackt. Laut Turbo-Handbuch (Abschnitt D.6) packt Turbo-Pascal Felder zwar automatisch; aber (wie leider erst wieder ein entsprechender Test zeigt) offenbar nicht mit dem Bit sondern mit dem Byte als kleinster Packungseinheit. Die in UCSD-Pascal praktikable Definition von real_maske nimmt in Turbo-Pascal 48 Bytes in Anspruch und ist somit zur Aufdeckung der Bitstruktur von 6-Byte-Gleitkommazahlen untauglich.

4. Es soll hier nicht der Eindruck erweckt werden, daß das Turbo-Pascal-Handbuch besonders schlecht ist; im Gegenteil, unter den mir bekannten Programmier-Handbüchern gehört es mit Abstand zur Gruppe der besten, und auch seine Beschreibung des Gleitkommaformats ist noch vergleichsweise ausführlich. Andererseits zeigt diese Bemerkung aber auch, wie schlecht es im allgemeinen um Programmier-Handbücher bestellt ist.

5. In der Informatik wird allgemein viel Aufwand betrieben, um zu gewährleisten, daß erstellte Programme korrekt sind. Im Lichte dieses Aufwands erscheint es mir als inkonsequent, welche Fehlerquellen man im Bereiche der Gleitkommaarithmetik hinzunehmen bereit ist; ja, daß im allgemeinen nicht einmal auf die Existenz und die Art dieser Fehlerquellen hingewiesen wird. Ein erster Schritt, der Fehlerquelle „Gleitkommaarithmetik" Rechnung zu tragen, muß m. E. darin bestehen, die Struktur dieser Gleitkommadarstellung transparent zu machen. Dies sollte allerdings bereits von der Herstellerseite her geschehen und nicht erst durch eine „Versuch-und-Irrtum"-Strategie der Benutzer.

Literatur:

[1] *Borland International: Turbo-Pascal-Version 3.0.* Heimsoeth, München, 1985.
[2] *Grözinger, M.:* Der gläserne Apfel. K. Stimmler Elektronik, Tübingen, 1986.
[3] *Kaiser, R.:* Grundlegende Elemente des Programmierens. Birkhäuser Verlag, Basel, 1985.

Zitat des Monats

Hinweis

Ein Konzept für die Datensicherung auf Ihrem PC könnte z. B. folgendermaßen aussehen:

Sie haben die Funktion s – Systemabzug auf MB-Kassette schreiben – gewählt.

ACHTUNG: Verwenden Sie diese Funktion maximal 1mal im Monat, da eine zu häufige Anwendung das Laufwerk beschädigen kann.

Sinix-Betriebssystem-Handbuch, Sinix V2.0, Band 2, Seite 5–75
gefunden von Patrik Fäh, Freiburg/Schweiz.

Josef Ellenrieder

Datenbank-systeme

Teil 1: Wichtige Begriffe anschaulich erklärt

Das Aufbewahren von Informationen, die man irgendwann gezielt wiederfinden will, gehört zu den traditionellen Aufgaben des Computers. PCs bieten heute vielfältige, kaum überschaubare Alternativen, Informationen mit Datenbanken zu verwalten. Diese dreiteilige Artikelserie klärt zunächst einige wichtige Begriffe, versucht dann die Frage zu beantworten, welches Datenbanksystem das richtige ist, und beschäftigt sich schließlich im letzten Teil mit dem Zugang zu großen Datenbanken.

Es ist erstaunlich, daß die Fachliteratur keine eindeutige und exakte Definition für den Begriff „Datenbanksystem" bereithält. Wir erklären ihn an einem Beispiel und betrachten dazu verschiedene Abteilungen eines Geldinstituts: Spar-, Kredit- und Wertpapierabteilung. Jede Abteilung führt eine Kartei über ihre Kunden und deren Konten (*Bild 1*).
In den Abteilungen der Bank wird nun die EDV eingeführt. Eine Lösungsmöglichkeit wäre es, die Dateien für die jeweiligen Programme – Sparkontenverwaltung, Kreditverwaltung und Wertpapierkontenverwaltung – entsprechend den bestehenden Karteien zu organisieren. Jedes Programm hat dann seine eigenen Daten zur Verfügung (*Bild 2*).

Unabhängige Datenverwaltung

Diese Lösung hat aber verschiedene Nachteile. Viele Kunden haben ein Sparkonto und Wertpapiere oder gleichzeitig einen Kredit. Ihre Namen und Adressen sind dann sowohl in der Sparkontendatei als auch in der Wertpapierkontendatei oder der Kreditdatei gespeichert. Ändert sich nun die Adresse eines solchen Kunden, muß die Änderung in allen betroffenen Dateien durchgeführt werden. Dies bedeutet einen hohen Bedarf an Speicherplatz und großen Änderungsaufwand. Will die Kreditabteilung Informationen über Spar- oder Wertpapierkonten ihrer Kreditnehmer haben, müssen zu den betroffenen Dateien entweder neue Zugriffswege – sozusagen Querverbindungen – geschaffen oder für die Kreditverwaltung eine eigene Spar- oder Wertpapierkonten-Datei eingerichtet werden. Aus diesen Gründen dürfte die Lösung gemäß *Bild 3* mit einem Datenbanksystem günstiger sein. Ein Datenbanksystem versucht die wiederholte physische Abspeicherung des gleichen Datenbestandes (die sogenannte Daten-Redundanz) zu vermeiden.
In unserem Fall handelt es sich um einen allgemeinen Kundenbestand mit allen kundenspezifischen Informationen (Name, Adresse, Geburtsdatum, Beruf usw.). Auskunft über die einzelnen Konten und Informationen über ihre Inhaber kann über Kundennummern oder -namen erteilt werden. Die Kundendaten sind nur noch einmal im System gespeichert. Wechselt ein Kunde seine Adresse, so genügt es, die neue Anschrift nur einmal zu erfassen. Sie steht dann sofort allen Programmen zur Verfügung.
Die Programme nehmen nicht mehr direkt Verbindung zu den Daten auf, sondern die benötigte Datenmenge wird über das Datenbanksystem bereitgestellt. Die Verwaltung der Daten kann unabhängig von den Anwendungen erfolgen. So können jederzeit neue Aspekte hinzugefügt werden (z. B. der zweite Wohnsitz eines Kunden), ohne daß in den bestehenden Anwendungsprogrammen Änderungen vorgenommen werden müssen. Modifikationen sind nur da notwendig, wo der neue Aspekt direkt verarbeitet werden soll. Außerdem kann auch die physikalische Ablageform des Verwaltungsteils geändert werden, ohne daß Änderungen der Anwendungsprogramme nötig sind.
Dies nennt man logische und physikalische Unabhängigkeit der Daten. Logische Unabhängigkeit bedeutet, daß die Daten in ihrer Darstellung gegenüber dem Anwendungsprogramm unabhängig von der physikalischen Abspeicherungsart sind. Die Sicht der Daten durch die Anwendung kann bedeutend einfacher sein, als die tatsächliche Form der abgespeicherten Daten. Physikalische

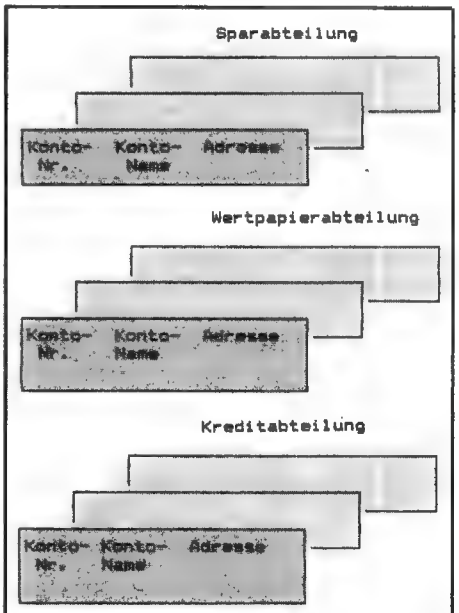


Bild 1. Datenverwaltung ohne EDV: Jede Abteilung führt eine eigene Datei

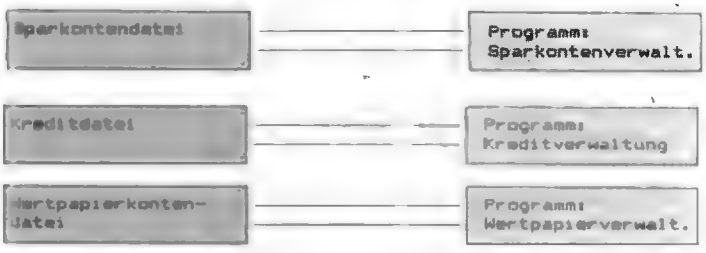


Bild 2. Eine schlechte Lösung: Jedem Programm ist eine eigene Datei zugeordnet

Datenunabhängigkeit ist die Möglichkeit, in der physikalischen Speicherungsform der Daten auf dem Datenträger Änderungen vorzunehmen, ohne auch die Anwendungsprogramme ändern zu müssen.
Zusammenfassend kann man sagen: In einem Datenbanksystem werden inhaltlich zusammenhängende Daten mit möglichst geringer Redundanz – wiederholter physikalischer Abspeicherung – unabhängig von den Programmen, die sie benutzen, gespeichert. Den Anwendungsprogrammen wird vom Datenbanksystem die angeforderte Teilmenge an Daten in der vom Programm benötigten Form zur Verfügung gestellt. Ein gutes Datenbanksystem garantiert eine möglichst hohe physikalische und logische Datenunabhängigkeit.

Datenbank und Datenbanksystem

Ein Datenbanksystem besteht in der Regel aus mehreren Datenbanken und dem entsprechenden Verwaltungsteil. Eine einzelne Datenbank umfaßt eine bestimmte Menge von Daten. In unserem Beispiel könnten in Bild 3 die Kundendaten in einer eigenen Datenbank innerhalb des Gesamtsystems liegen. In der EDV werden logisch zusammengehörende Daten als Einheit betrachtet. Dies gilt auch für Datenbanken. In unserem Beispiel kann die gesamte Information über einen bestimmten Kunden als Informationseinheit (Satz) betrachtet werden. Die einzelnen Aspekte, z. B. der Gesamtumsatz, werden als Datenfelder bezeichnet. Ebenso kann die Information über ein Kundenkonto als Informationseinheit (Satz) behandelt werden.
Der Aspekt (das Feld), der eine Informationseinheit eindeutig identifiziert, heißt Schlüssel. In unserem Sparkontenbeispiel kann die gesamte Information über ein bestimmtes Konto (Kundennummer, Guthaben, Zinssatz) als Informationseinheit (Satz) betrachtet werden. Die einzelnen Aspekte, z. B. Laufzeit, bezeichnet man als Datenfelder.
Ebenso kann die Information über einen bestimmten Kredit als Informationseinheit behandelt werden. Der Aspekt (Feld), welcher eine Informationseinheit (Satz) eindeutig identifiziert, heißt Schlüssel. In unserem Beispiel haben wir drei verschiedene Typen von Informationen. Die Bezeichnung Informationstyp soll die allgemeine Darstellung, die Bezeichnung Informationseinheit eine konkrete Erscheinungsform umfassen. Eine Informationseinheit ist die Menge aller Daten über den Kunden mit der Kundennummer „01234". Die Informationseinheit wird durch den Schlüssel Kundennummer, der hier den Wert „01234" hat, identifiziert.

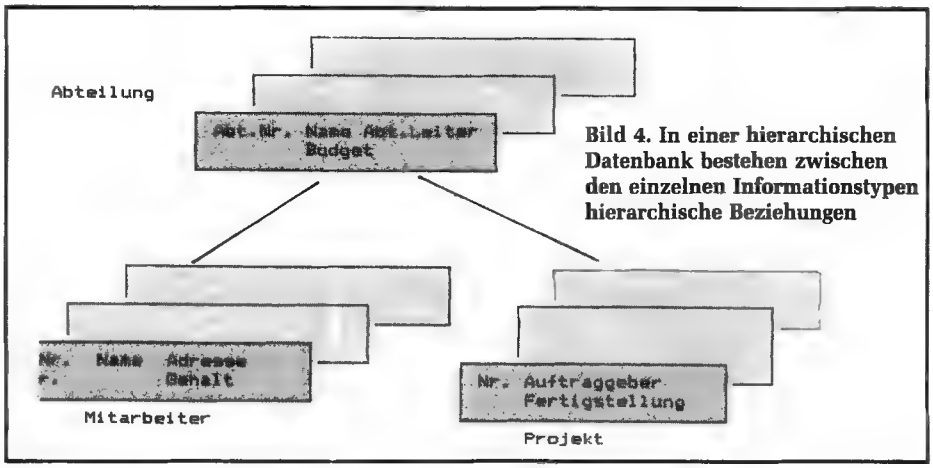


Bild 4. In einer hierarchischen Datenbank bestehen zwischen den einzelnen Informationstypen hierarchische Beziehungen

Informationstypen und ihre Beziehungen

Die verschiedenen Datenbanksysteme unterscheiden sich nach der Art und Weise, wie die Beziehungen zwischen den Informationstypen strukturiert sind. Die wichtigsten Datenbankmodelle sind
– das hierarchische Modell,
– das Netzwerk-Modell,
– das Relationsmodell.
Natürlich gibt es auch Datenbanksysteme, denen eine Mischform zugrundeliegt. In einer hierarchischen Datenbank bestehen zwischen den einzelnen Informationstypen hierarchische Beziehungen (*Bild 4*). Es gibt drei Informationstypen: Abteilung, Mitarbeiter und Projekt. Hinter jedem steht eine bestimmte Menge von konkreten Informationseinheiten. Die Identifikationsfelder (Schlüssel) sind Abteilungs-, Mitarbeiter- oder Projektnummern.
Die hierarchische Beziehung kommt darin zum Ausdruck, daß die Informationstypen „Mitarbeiter" und „Projekte" abhängig vom Informationstyp „Abteilung" sind. Es wäre natürlich auch eine umfangreichere Verzweigung denkbar. So könnte der Informationstyp „Kinder", der Daten über die Kinder der Mitarbeiter umfaßt, noch vom Informationstyp „Mitarbeiter" abhängig sein.
Der Informationstyp der höchsten Ebene wird als Wurzel der Struktur bezeichnet. Die Abhängigkeit äußert sich darin, daß ein Zugriff nur über die Wurzel mit anschließender Abarbeitung der abhängigen Typen möglich ist. Möchte man etwa das Gehalt des Mitarbeits „002" in der Abteilung „XY" wissen, muß zuerst mit Hilfe des Schlüssels, der hier den Wert „XY" hat, die richtige Informationseinheit in der Wurzel „Abteilung" angesprochen werden. Von dieser Informationseinheit aus ist dann der Zugriff auf den ersten Mitarbeiter der Abteilung XY möglich.
Die einzelnen Mitarbeiter müssen der Reihe nach durchsucht werden, bis mit dem Schlüssel „002" der richtige identifiziert ist. Die gestrichelten Linien in *Bild 5* sollen die Vorgehensweise noch einmal verdeutlichen.
Über ein zusätzliches Inhaltsverzeichnis (Index), den Sekundärindex, kann man

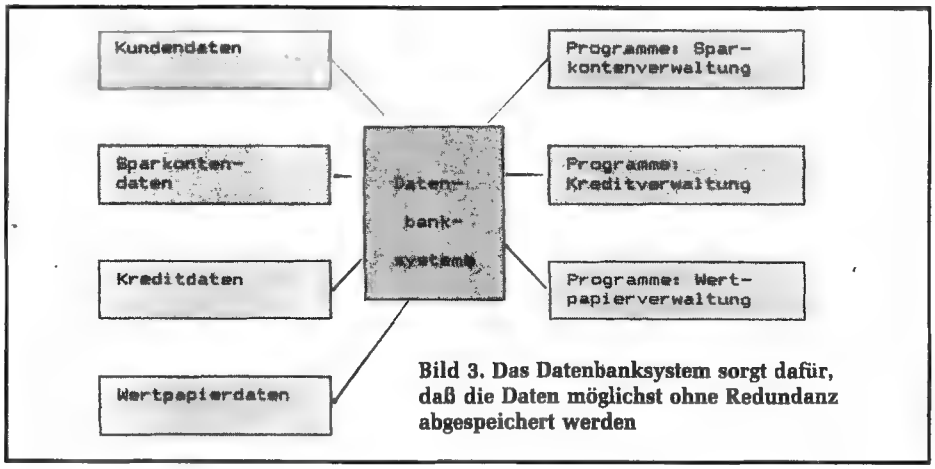


Bild 3. Das Datenbanksystem sorgt dafür, daß die Daten möglichst ohne Redundanz abgespeichert werden

abhängige Informationseinheiten direkt ansprechen. In *Bild 6* umfaßt der Index in sortierter Reihenfolge die Mitarbeiternummern und über Adreßzeiger den Verweis zur entsprechenden Mitarbeiter-Informationseinheit.

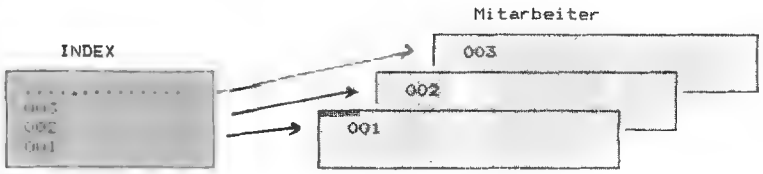


Bild 6. Direktsuche per Index: Der Index umfaßt hier in sortierter Reihenfolge die Mitarbeiternummern und über Adreßzeiger den Verweis zur entsprechenden Mitarbeiter-Informationseinheit

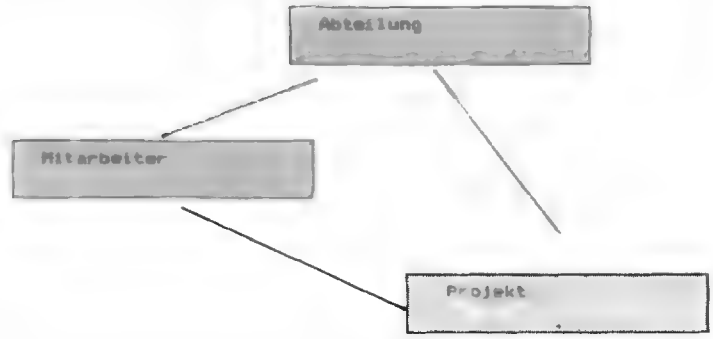


Bild 7. Erweitert man das Beispiel von Bild 4 so, daß ein Mitarbeiter auch einem bestimmten Projekt zugeordnet sein kann und die Beziehungen zwischen den Informationstypen nicht mehr nur in eine Richtung gehen, dann erhält man eine Datenbank nach dem Netzwerkmodell

Hierarchie in der Datenbank

Eine wichtige Grundidee bei hierarchisch strukturierten Datenbankmodellen ist es, eine Informationseinheit mit allen abhängigen Einheiten physikalisch als Ganzes abzulegen. So ist der Zugriffsweg für bestimmte Anwendungen optimal, für andere aber weniger günstig. Innerhalb einer Datenbank ist nur eine hierarchische Struktur möglich. Durch Definition logischer Abhängigkeiten und Verknüpfung mehrerer hierarchischer Datenbanken können auf Umwegen auch Netzstrukturen realisiert werden. Das hierarchische Datenbankmodell ist das älteste der drei genannten Modelle. Eines der bekanntesten Modelle ist im Großrechnerbereich „IMS" von IBM. Im Bereich des Personal Computers gibt es aufgrund des relativ hohen Verwaltungsaufwandes, der geringen Flexibilität und der komplizierten Zugriffsweise keine hierarchischen Datenbanksysteme.

Beim hierarchischen Modell darf jede abhängige Informationseinheit genau einen Vorgänger haben. Beim Netzwerkmodell dagegen kann jede Informationseinheit mehrere übergeordnete Informationseinheiten besitzen. Ist beim hierarchischen Modell die Beziehung immer von der übergeordneten zur abhängigen Einheit gerichtet, so kann sie bei der Netzwerkstruktur beidseitig auftreten. Die Zahl der Beziehungen zwischen den einzelnen Netzwerkelementen ist nicht beschränkt.

Bild 7 zeigt das Beispiel aus Bild 4, erweitert um den Sachverhalt, daß ein Mitarbeiter an einem bestimmten Projekt arbeitet. Die Beziehungen zwischen den Informationstypen sind nicht mehr ein-, sondern beidseitig. Natürlich können, wie Bild 8 zeigt, Netzwerkstrukturen bedeutend komplexer sein.

Gleichberechtigung im Netzwerk

Das Netzwerkmodell könnte man auch als die Verallgemeinerung des hierarchischen Modells bezeichnen. Wichtig ist, daß alle Informationstypen gleichberechtigt sind und deswegen unabhängig voneinander in der jeweils sinnvollsten Art gespeichert werden können. Die definierten Beziehungen zwischen den Informationseinheiten werden durch Querverweise (Zeiger) verwirklicht. Zum Beispiel ist bei einer bestimmten Informationseinheit des Typs „Mitarbeiter" physikalisch die Adresse abgelegt, wo die zugehörigen Informationseinheiten des Typs „Abteilung" und „Projekt" zu finden sind. Diese Unabhängigkeit ermöglicht es, je nach den Gegebenheiten die günstigste Speicherorganisation zu wählen.

Der Zugriff auf die Daten kann unterschiedlich erfolgen: Man kann eine Informationseinheit direkt ansprechen oder die Pfade zwischen den Elementen in der Struktur beliebig auf- und abwärts verfolgen. Die Identifikation der Informationseinheiten erfolgt durch den Schlüssel. Im Beispiel können die Daten des Mitarbeiters „002" direkt abgerufen werden, ohne den Weg über die Wurzel nehmen zu müssen. Andererseits könnte man auf einem bestimmten Projekt aufsetzen und von dort aus zu den einzelnen am Projekt beteiligten Mitarbeitern steuern.

Der Grad der Realisierung des Netzwerk-Datenbanksystems ist unterschiedlich. Im PC-Bereich sind diese Datenbanksysteme sehr selten, da sie in der Regel nur von Spezialisten effektiv genutzt werden können. Als Beispiel kann MDBS genannt werden. Die im PC-Bereich dominierenden Datenbanksysteme sind relational. Das Relationenmodell beruht auf einer mit Hilfe der Relatio-

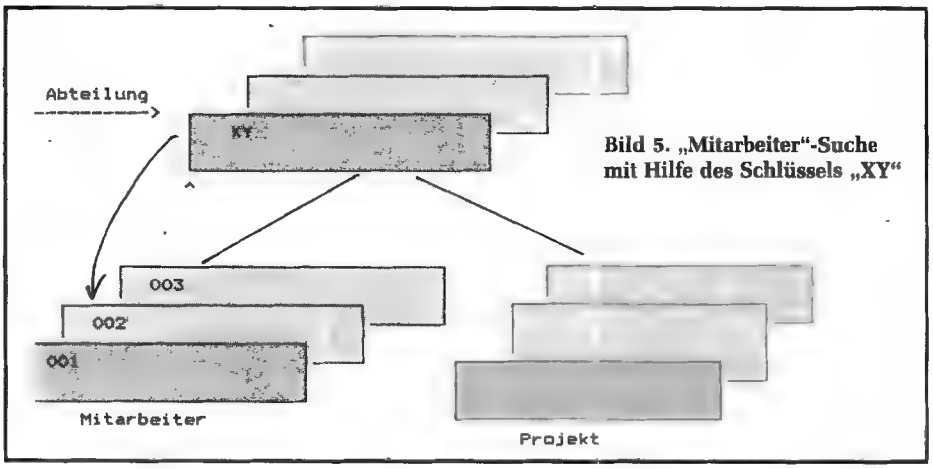


Bild 5. „Mitarbeiter"-Suche mit Hilfe des Schlüssels „XY"

nen-Algebra dargestellten Methodik, wobei hier auf die mathematische Formulierung verzichtet werden kann.

Tabellen-Datendarstellung:

Die verständlichste Datendarstellung für den Nicht-Fachmann ist die zweidimensionale Tabelle (*Bild 9*). Tabellen sind eine vertraute Darstellungsform und bleiben gut im Gedächtnis. Sie sind, mathematisch betrachtet, Relationen. Eine relationale Datenbank besteht aus einer Menge solcher Relationen.
Eine Zeile in einer Tabelle wird auch als „Tupel" bezeichnet. Eine Zeile umfaßt die auf ein bestimmtes Objekt bezogenen Datenwerte. Die Spalte in einer Tabelle nennt man auch Attribut. Die Menge aller Werte, welche eine Spalte annehmen kann, heißt Domäne dieser Spalte.
Im Beispiel haben wir die Relation mit dem Namen „Abteilung". Dort gibt es die Spalte Abteilungsnummer, Name, Abteilungsleiter, Budget. Die Domäne für das Attribut Abteilungsleiter umfaßt mögliche Familiennamen. Man kann bei der Definition einschränken, daß die Domäne nur die Namen aller im Unternehmen tätigen Abteilungsleiter, die Namen aller Mitarbeiter oder alle möglichen Familiennamen umfassen darf. In den Zeilen unseres Beispiels hat die Spalte „Abteilungsleiter" die Werte „Huber", „Bauer", „König" angenommen. Das wesentliche an diesen Tabellen ist, daß sie keine Struktur aufweisen dürfen: keine geschachtelten Teile, keine abhängigen Teile usw. Diese Strukturlosigkeit erlaubt es, daß die physikalische Speicherungsform frei und unabhängig von den logischen Zusammenhängen ist. Es ist auch nicht nötig, vordefinierte Zugriffspfade anzulegen.

Direktes Anfragen beim Relationenmodell

Wie werden nun aber Zusammenhänge wie in *Bild 7* (erweiterte Hierarchie) und die entsprechenden Beziehungen verwirklicht? Jede Netz- oder Baumstruktur kann auf eine Menge von Relationen zurückgeführt werden. Die Verknüpfung erfolgt durch Felder gleichen Inhalts. Sie ist aber erst vorhanden, wenn die konkrete Abfrage erfolgt. Die Position des Feldes in der Tabelle und Adressen spielen keine Rollen. Dies ermöglicht es, nach allen Feldinhalten zu suchen.
Hier einige mögliche Fragestellungen zur Datenbank in *Bild 9* und *Bild 10*:
– Liste aller Mitarbeiter mit einem Gehalt von mehr als 5000 DM.
– Liste aller Mitarbeiter der Abteilung XY, die an einem Projekt arbeiten, das vor 1986 fertiggestellt werden muß.

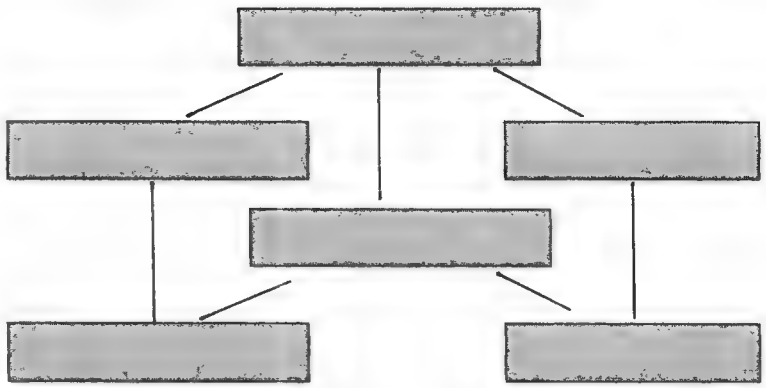


Bild 8. Beispiel für die Beziehungen in einer „Netzwerk"-Datenbank

Antworten: Mitarbeiter Huber, König, Meier, Hofer.
Die Verbindung zwischen den Relationen erfolgt in der letzten Frage über den Inhalt des Feldes „Projektnummer". Diese Möglichkeit der Fragestellung ist der Grund, daß bei relationalen Datenbanksystemen Endbenutzer auch Anfragen direkt (interaktiv) bearbeiten können. Dazu ist eine geeignete Abfragesprache nötig. Bei strukturierten Datenbanken muß eine Fragestellung erst in ein Programm umgesetzt werden; Abfragesprachen ermöglichen es auch Benutzern ohne Datenbankerfahrung, relativ schnell zu Ergebnissen zu kommen.

Zugriff per Indizes

Änderungen in der Organisationsform auf dem Speichermedium müssen so vorgenommen werden können, daß auf die Relationen und ihre Auswertungsmöglichkeiten keine Auswirkungen entstehen. Das Relationenmodell trifft keine Aussage über die physikalische Ablage und Verwaltung der Daten. Hier ist nicht so sehr die Speicherungsform von Bedeutung, vielmehr die Zugriffsart. Bei der heute verfügbaren Hardware ist die praktischste Lösung die Führung von Indizes (Inhaltsverzeichnissen). Diese Methode sucht mit einem Kriterium in einer Tabelle, geht aufgrund des Verweisens in der Tabelle zu der entsprechenden Datenzeile und vergleicht in der gefundenen Zeile die weiteren Suchkriterien mit den vorhandenen Feldwerten. Natürlich besteht die Gefahr, daß durch das Suchen in der Tabelle der Zugriff auf die Daten besonders bei großen Mengen langsam wird. Dem soll durch neue Hardware-Techniken abgeholfen werden. Besonders entscheidend ist hier die Entwicklung auf dem Gebiet der Assoziativspeicher.

Abt.Nr.	Name	Abt.Leiter	Budget
20	Personal	Huber	300.000 DM
30	Service	König	200.000 DM
40	Einkauf	Bauer	800.000 DM
50	Verkauf	Schön	750.000 DM

Bild 9. Eine für den Nicht-Fachmann verständliche Darstellungsform: die Tabelle

Nr.	Name:	Adresse:	Gehalt:	Abt.Nr.	Proj.Nr.
1	Huber		7.000	20	11
2	König		5.500	30	12
3	Bauer		8.000	50	13
4	Maier		3.000	40	11
5	Hofer		4.000	55	16
6	Baum		6.000	10	12

Projekt.Nr.	Titel	Fertigungstellung:	Auftraggeber:
11	Organis. Textverarb.	31.12.86	Sekretariat 1
12	Formularanpassungen	31.03.87	ORG
13	Einführung PC-Center	30.06.87	Rechenzentrum

Bild 10. Zwei weitere Tabellen, die sich aus der Datenbank von Bild 9 ergeben

Michael Franz

Hinter den Kulissen von Pascal

Pascal gehört neben C sicher zu den am besten strukturierten Programmiersprachen. Wie diese Strukturen von einem Compiler auf den nicht strukturierten Befehlssatz eines Prozessors übersetzt werden, wird hier anhand des 68000-Prozessors gezeigt.

Der hier beschriebene Pascal-F-Compiler hat die schöne Eigenschaft, als Output den Code in 68000-Mnemonics zu liefern. Daher kann man das Ergebnis noch in Klarschrift lesen. Aber auch wer mit diesem Maschinencode noch nicht vertraut ist, kann die beschriebenen Prinzipien leicht verstehen.
Das folgende Programmbeispiel ist das kürzeste, korrekte Pascal-F-Programm; es tut überhaupt nichts:

```
PROGRAMM (INPUT, OUTPUT);
BEGIN
END.
```

Pascal-Kennern wird aufgefallen sein, daß der Programmname fehlt, was aber ohne Belang ist. Dieses Programm macht eigentlich nichts – und doch erzeugt der Compiler schon eine ganze Menge: Zunächst einmal werden eine Reihe von Einsprüngen definiert, welche das fertige Programm vom Betriebssystem und einem sogenannten Laufzeitmodul erwartet (*Bild 1*). Umfangreichere Programmteile, wie z. B. eine Gleitpunkt-Operation werden nämlich nicht jedesmal in den Code einkopiert, sondern als Unterprogramm aus dem Laufzeitmodul aufgerufen. Im Programmcode stehen also nur noch der Unterprogrammaufruf und evtl. eine Fehlerprüfung.
Bei den weiteren Beispielen wurde diese Definitionsleiste jeweils weggelassen.

```
ORG $030000
OFFSET $00000000

ENTRY EQU $20024
MULS32 EQU ENTRY+0*4
DIVS32 EQU ENTRY+1*4
ADDF32 EQU ENTRY+2*4
SUBF32 EQU ENTRY+3*4
MULF32 EQU ENTRY+4*4
DIVF32 EQU ENTRY+5*4
RELATION EQU ENTRY+6*4
CNVRTIF EQU ENTRY+7*4
CNVRTFI EQU ENTRY+8*4
NUOUT EQU ENTRY+9*4
FNUOUT EQU ENTRY+10*4

TXTDIR EQU ENTRY+11*4
TXTAUS EQU ENTRY+12*4
OVERR EQU ENTRY+13*4
DIV0ERR EQU ENTRY+14*4
IDXERR EQU ENTRY+15*4
CRLF EQU ENTRY+16*4
READINT EQU ENTRY+17*4
SINUS EQU ENTRY+18*4
LAB0 EQU SINUS
COSINUS EQU ENTRY+19*4
LAB1 EQU COSINUS
SQRT EQU ENTRY+20*4
LAB2 EQU SORT

   ;hier beginnt der eigentliche Code
MOVEM.L D0–D7/A0–A6,–(A7)
MOVE.L A7, SAVEA7
LEA.L DATA,A0
MOVEA.L A0,A3
JMP LAB3
LAB3:

BRK: MOVEA.L SAVEA7,A7
MOVEM.L (A7)+,D0–D7/A0–A6
RTS
DC.B ,VERSION 1.0 11/86'
DS.W 0
FREE EQU *

   ;ab hier RAM nötig!!
EABUFF EQU FREE
IOSTAT EQU EABUFF+84
SAVEA7 EQU IOSTAT+2
RF EQU SAVEA7+4
DATA EQU RF+4*20
END
```

Bild 1. Soviel Assemblercode erzeugt ein „leeres" Programm

```
   :
   :
MOVEM.L D0–D7/A0–A6,–(A7)
MOVE.L A7,SAVEA7
LEA.L DATA,A0
MOVEA.L A0,A3
JMP LAB3
LAB3:
MOVE.L 4(A3),D0     ;Operand B holen
MOVE.L D0,0(A3)     ;an A zuweisen

BRK: MOVEA.L SAVEA7,A7
MOVEM.L (A7)+,D0–D7/A0–A6
RTS
   :
   :
```

Bild 2. Der Compiler hat eine einfache Zuweisung übersetzt

Der Programmcode beginnt also erst weiter unten und sorgt hier nur dafür, daß die Rücksprungadresse auch in jedem Fehlerfall wiedergefunden werden kann.
Bei Programmende wird das Label BRK erreicht, wo der Rücksprung (in das aufrufende Betriebssystem) durchgeführt wird.
Auch wenn in diesem kurzen Programm noch keine Variablen verwendet wurden, sind doch einige Bereiche schon für diesen Zweck vorbereitet: ab Adresse FREE muß RAM verfügbar sein (wichtig, wenn das fertige Programm z. B. in einem EPROM ablaufen soll) und ab der Adresse Data werden die eigentlichen Pascal-Variablen angelegt.
Ein solcher Befehl wird in Pascal-F wie folgt geschrieben, wobei die Variablen unbedingt vorher definiert sein müssen:

```
PROGRAM (INPUT,OUTPUT);
VAR A,B :INTEGER;
BEGIN
A:=B
END.
```

Die eigentliche Zuweisung besteht also nur aus zwei Maschinenbefehlen und ist daher ungeheuer schnell. Für die Adressierung der beiden Variablen muß man folgendes wissen: Das Adreßregister A3 zeigt immer auf den Anfang des aktuellen Variablenbereiches (warum nicht absolut adressiert werden kann erkennt man bei Funktionen und Prozeduren). Ein INTEGER (= Ganzzahl) ist außerdem mit einer Länge von 4 Bytes definiert, so daß die erste Variable unter der Adresse in A3, die zweite eben 4 Bytes weiter zu finden ist (*Bild 2*). Bei der Verwendung von REAL-Variablen werden entsprechend jeweils 6 Bytes Speicher reserviert.

Die Arithmetik

Natürlich muß der Compiler bei der Auswertung eines arithmetischen Aus-

drucks die Prioritäten der einzelnen Operationen beachten. Mit absteigender Priorität werden ausgeführt:

(.....)	explizite Klammerung
–	unäres Minus
* /	Multiplikation, Division
+,–	Addition, Subtraktion

```
PROGRAM (INPUT,OUTPUT);

VAR A,B,C,D :INTEGER;
BEGIN
A:=B+C*(D–1)
END.
```

Das Beispiel (*Bild 3*) ist so gewählt, daß die Auswertung von hinten her erfolgen muß, was das Programm ja auch ordentlich macht.
Bei der Auswertung solcher Ausdrücke (in der Sprache der Compilerspezialisten sind es Expressions) treten oft Zwischenergebnisse auf, die erst später weiter verwendet werden. Man darf sie nicht in den Prozessor-Registern ablegen, weil sie dort z. B. bei der Ausführung anderer Rechenoperationen zerstört würden. Das Programm benötigt für die Pufferung also zusätzlichen Speicher, der vom Programmierer in Pascal nicht definiert wurde.
Dabei ist schon zur Übersetzungszeit bekannt, wie viele Puffer für die Auswertung einer einzelnen Expression erforderlich sind. Der Compiler legt solche Puffer nun direkt hinter den vom Benutzer definierten Variablen an und adressiert sie auch genauso. Aber es besteht ein wesentlicher Unterschied zwischen Pascal-Variablen und Hilfszellen: Die Hilfszellen sind nur kurzzeitig nötig und ihr Inhalt ist nach Abarbeitung der Expression ohne Bedeutung. Wenn später wiederum Hilfszellen gebraucht werden, dürfen problemlos dieselben Speicherplätze wieder verwendet werden. Anwender-definierte Variable dürfen aber nur dann beschrieben werden, wenn es explizit durch eine Zuweisung verlangt wird.
Wie man sieht, sind für diese eine Zeile in Pascal eine ganze Reihe von Assembler-Statements erforderlich. Dabei kommt es vor, daß sich vielleicht dieser oder jener Befehl einsparen ließe, wenn man direkt in Assembler programmiert. Das liegt daran, daß der Compiler einen universellen Code zur Auswertung von Expressions generiert, der nach einem vorgegebenen Schema erzeugt wird. Je flexibler dieses Schema, desto besser (aber auch desto aufwendiger) der Compiler.

Programm am Scheideweg

Eines der wichtigsten Elemente einer Programmiersprache ist die Verzweigung und die bedingte Ausführung von bestimmten Programmteilen. Ein auch in allen anderen Sprachen vorhandenes Element dazu ist die IF...-Anweisung:

```
PROGRAM (INPUT,OUTPUT);

VAR A,B,C,D :INTEGER;
BEGIN
IF A=1
   THEN
      B:=1
   ELSE
      BEGIN
      B:=2
      END
END.
```

Diese strukturierte Anweisung muß nun zerlegt werden, wozu auf Assembler-Ebene nur noch (unstrukturierte) Sprünge zur Verfügung stehen (*Bild 4*). Außerdem müssen die erforderlichen Labels (Sprungmarken) erzeugt werden. Dazu wird vom Compiler ein einfacher Label-Zähler geführt.

```
if false then goto else-Zweig
   Befehle im then-Zweig
   goto next-statement

else-Zweig:
   Befehle im else-Zweig
next-statement:
```

Für ein IF-Statement sind zwei Labels erforderlich, wie man sieht. Zu Beginn des IF-Statements werden die beiden erforderlichen Nummern reserviert (auch wenn die Labels selbst erst später in den

```
   :
   :
MOVEM.L D0–D7/A0–A6,–(A7)
MOVE.L A7,SAVEA7
LEA.L DATA,A0
MOVEA.L A0,A3
JMP LAB3
LAB3:
   ;hier beginnt das eigentliche
   ;Hauptprogramm

   ;hier beginnt die Auswertung der Subtraktion
MOVE.L 12(A3),D0          ;lade D (die 3. definierte
                          ;Variable; 3*4 Bytes = 12)
MOVE.L #1,D2              ;lade den Direktoperanden
SUB.L D2,D0               ;führe schnelle 32-Bit-
                          ;Subtraktion aus
BVC.S *+2+6+6             ;springe, wenn kein Über-
                          ;lauf aufgetreten
JSR DVERR                 ;gib Fehlermeldung aus
JMP BRK                   ;und beende das Programm

MOVE.L D0,16(A3)          ;lege das Ergebnis der Sub-
                          ;traktion auf Hilfszelle 1

   ;hier beginnt die Auswertung der Multiplikation
MOVE.L 8(A3),D0           ;lade C (die 2. definierte
                          ;Variable 2*4 Bytes = 8)
MOVE.L 16(A3),D2          ;lade letztes Zwischenergebnis
                          ;von Hilfszeile 1
JSR MULS32                ;führe 32-Bit-Multiplikation
                          ;mit Vorzeichen aus
BCC.S *+2+6               ;springe, wenn kein Überlauf
JMP BRK                   ;beende das Programm bei Fehler;
                          ;die Fehlermeldung wurde
                          ;schon von MULS32 gegeben
MOVE.L D0,20(A3)          ;speichere das Ergebnis der
                          ;Multiplikation auf Hilfszelle 2

   ;hier beginnt die Auswertung der Addition
MOVE.L 4(A3),D0           ;lade C (die 1. definierte
                          ;Variable; 1*4 Bytes = 4)
MOVE.L 20(A3),D2          ;lade Ergebnis der Multiplikation
                          ;aus Hilfszeile 2
ADD.L D2,D0               ;führe schnelle 32-Bit-
                          ;Addition aus
BVC.S *+2+6+6             ;springe im Normalfall
JSR OVERR                 ;Fehlerfall wie bei Sub.
JMP BRK
MOVE.L D0,24(A3)          ;speichere Ergebnis auf Hilfszelle 3
MOVE.L 24(A3),D0          ;führe eine Zuweisung von der
MOVE.L D0,0(A3)           ;Hilfszelle 3 an die Variable 0
                          ;d. h. A durch
BRK: MOVEA.L SAVEA7,A7    ;Programmende
MOVEM.L (A7)+,D0–D7/A0–A5
RTS
   :
   :
```

Bild 3. Ein mathematischer Ausdruck wurde compiliert

```
   :
   :
MOVEM.L D0–D7/A0–A6,–(A7)
MOVE.L A7,SAVEA7
LEA.L DATA,A0
MOVEA.L A0,A3
JMP LAB3
LAB3:

   ;hier beginnt das Hauptprogramm
MOVE.L 0(A3),D0          ;lade Variable A
MOVE.L #1,D2             ;lade Vergleichswert
MOVEQ.L #–1,D4           ;besetze D4 als Flag-Register
                         ;mit true (–1)
CMP.L D2,D0              ;vergleiche
BEQ.S *+4                ;springe, wenn Vergleichs-
                         ;bedingung erfüllt
CLR.L D4                 ;setze Flag auf false
MOVE.L D4,16(A3)         ;Vergleichsergebnis auf
                         ;Hilfszelle
BEQ LAB4                 ;springe, wenn IF-Bedingung
                         ;falsch ist
   ;hier beginnt der THEN-Zweig
MOVE.L #1,D0
MOVE.L D0,4(A3)
JMP LAB5                 ;überspringe den ELSE-Zweig

   ;hier beginnt der ELSE-Zweig
LAB4:
MOVE.L #2,D0
MOVE.L D0,4(A3)

   ;gemeinsames Ende des IF-statements
LAB5:

BRK: MOVEA.L SAVEA7,A7
MOVEM.L (A7)+,D0–D7/A0–A6
RTS
   :
   :
```

Bild 4. Die IF...THEN...ELSE-Verzweigung in 68000-Code

Code eingesetzt werden können) und der Zähler weitergeschaltet. Wenn jetzt z. B. im Then-Zweig ein weiteres IF-Statement steht, kann es vorkommen, daß die hier vergebenen höheren Labelnummern im Assemblercode vor den Labelnummern des umfassenden IF-Statements stehen.
Wie im Assemblercode zu erkennen, sind der THEN- und der ELSE-Zweig vollkommen identisch aufgebaut. Die (hier überflüssige) Klammerung des ELSE-Zweiges durch BEGIN und END hat überhaupt keinen Einfluß auf den erzeugten Code.

Das Array

Eine für viele Fälle günstige Datenstruktur ist das Array. Während es dem Anwender (zumindestens in ein- und zweidimensionaler Form) sehr anschaulich vorkommt, bedeutet es für einen Compiler wirklich harte Arbeit, aus den angegebenen Indizes einen physikalischen Speicherplatz zu berechnen. Schließlich erwarten wir von dem fertigen Programm, daß es uns auf Fehler aufmerksam machen kann, wenn z. B. ein aktueller Index außerhalb der definierten Grenzen liegt:

```
PROGRAM (INPUT,OUTPUT);

VAR A :ARRAY[1..100,5..15]
OF INTEGER;
      B :INTEGER;

BEGIN
B:=A[10,5]
END.
```

Zu erkennen ist deutlich der Aufwand für die Fehlererkennung. Er ist praktisch genauso groß wie die Berechnung selber, aber was nützt ein Programm, welches unkontrollierte Aktionen durchführt, wenn aufgrund eines Programmierfehlers ein Index zu groß oder zu klein wird. Und solche Fehler kommen auch bei einem guten Programmierer vor. Interessant ist noch die Speicherverwaltung in diesem Beispiel (*Bild 5*). Das Array ist als erstes definiert und bekommt daher den Speicher ab der Relativadresse 0 zugewiesen. Die zweite definierte Variable, nämlich B, wird anschließend angelegt und hat, wie ersichtlich, die Relativadresse 4400. Das stimmt genau, denn das Array hat insgesamt 1100 Elemente zu je 4 Bytes, da es INTEGER-Werte enthält.

Unterprogramme

Eines der besonderen Kennzeichen von Pascal z. B. gegenüber Basic liegt in der Unterprogrammtechnik. In Pascal kann man auch innerhalb eines Unterprogramms lokale Variable definieren, welche nur solange existieren, bis das Unterprogramm wieder verlassen wird. Danach wird der hierfür belegte Speicher wieder freigegeben. Wenn nun beim Aufruf des Unterprogramms (z. B. durch das Hauptprogramm oder auch durch sich selbst bei Rekursion) schon eine Variable gleichen Namens existiert, wird diese sozusagen verdeckt (verschattet). Das Unterprogramm reserviert einen neuen Speicherplatz, arbeitet auf diesem und schaltet erst wieder auf die zuletzt definierte Variable zurück, wenn es verlassen wird. Dadurch können Unterprogramme arbeiten, ohne versehentlich die Daten des Hauptprogramms zu vernichten.
Jetzt ist auch zu verstehen, warum die Variablen im Assemblercode nicht absolut adressiert sein dürfen: für eine Variable desselben Namens würde immer derselbe physikalische Speicher verwendet – und das darf nicht sein, wie wir jetzt wissen.
Alle Variablen werden deshalb über ein Basisregister angesprochen. Wird ein Unterprogramm aufgerufen, wird das Basisregister auf einen freien Speicherbereich gesetzt. Neu definierte Variable werden somit automatisch in diesem freien Bereich angelegt.
Am Ende des Unterprogramms wird das Basisregister auf den Wert gesetzt, den es vorher hatte, so daß das aufrufende Programm korrekt mit seinen Variablen weiterarbeiten kann.

```
PROGRAM (INPUT,OUTPUT);

VAR A,B :INTEGER;

  PROCEDURE EX (VAR X, Y :
  INTEGER);
  (* VERTAUSCHT DIE WERTE VON
  2 INTEGER-VARIABLEN *)
  VAR A :INTEGER;
  (* LOKALE VARIABLE *)
  BEGIN
```

```
   :
   :
MOVEM.L D0–D7/A0–A6,–(A7)
MOVE.L A7,SAVEA7
LEA.L DATA,A0
MOVEA.L A0,A3
JMP LAB3

   ;hier beginnt das Hauptprogramm
LAB3:
CLR.L –(A7)             ;auf dieser Hilfszelle wird der
                        ;Index-Offset berechnet
MOVE.L #10,D0           ;lade aktuellen Index 1
CMP.L #1,D0             ;vergl. mit Index-Untergrenze
BLT.S *+2+6+2           ;ggf. springe zur Index-Fehler-Routine
CMP.L #100,D0           ;vergl. mit Index-Obergrenze
BLE.S *+2+6+6           ;ggf. überspringe Index-Fehler-Routine
JSR IDXERR              ;gib Fehlermeldung auf Monitor
JMP BRK                 ;beende Programm

   ;1. Index o.k
SUB.L #1,D0             ;subtrahiere Index-Untergrenze
MOVE.L #1,D1            ;benötigter Speicher bis zu
                        ;dieser Dimension (vorbesetzt mit 1)
MULU D1,D0
ADD.L (A7),D0           ;addiere zu bisherigem Offset
MOVE.L D0,(A7)          ;und speichere wieder

   ;hier beginnt die Auswertung der 2. Dimension
MOVE.L #5,D0            ;lade aktuellen Index-Wert
CMP.L #5,D0             ;vgl. mit Untergrenze
BLT.S *+2+6+2
CMP.L #15,D0            ;vgl. mit Obergrenze
BLE.S *+2+6+6
JSR IDXERR
JMP BRK
SUB.L #5,D0             ;subtrahiere Untergrenze der
                        ;2. Dimension
MOVE.L #100,D1          ;benötigter Speicher der
                        ;1. Dimension
MULU D1,D0
ADD.L (A7),D0           ;neuen Offset berechnen
MOVE.L D0,(A7)          ;und wieder speichern

   ;der Offset des adressierten Elementes ist berechnet
MOVE.L (A7)+,D7
LSL.L #2,D7             ;*4, da als INTEGER definiert
MOVE.L 0(A3,D7.L),D0    ;lade das adressierte ARRAY-Element
MOVE.L D0,4400(A3)      ;übergib an B
BRK: MOVEA.L SAVEA7,A7
MOVEM.L (A7)+,D0–D7/A0–A6
RTS
   :
   :
```

Bild 5. Arrays fordern dem Compiler einiges ab

```
A:=X; X:=Y; Y:=A
END;

BEGIN
EX(A,B)
END.
```

Nach diesem Blick hinter die Kulissen von Pascal wird zwar kaum jemand in der Lage sein, einen eigenen Compiler zu bauen. Aber da über Basic und sein inneres Arbeiten ausreichend veröffentlicht wurde, sollte hier die Wirkungsweise eines Compilers erläutert werden, denn auch ein intelligentes Programm braucht für niemanden nur als ‚black box' zu existieren.
Alle Beispiele wurden mit Pascal-F auf dem NDR-Computer übersetzt (nur die Kommentare wurden nachträglich hinzugefügt). Er arbeitet mit dem RDK-Grundprogramm zusammen und ist über den Elektronikladen zu beziehen.

Literatur:

Roland C. Backhouse: Syntax of Programming Languages (Prentice-Hall).
Goos, Hartmanis: Compiler Construction (Springer-Verlag).
M. Franz: Systembeschreibung zu Pascal-F auf dem NDR-Computer.

```
   :
   :
MOVEM.L D0–D7/A0–A6,–(A)
MOVE.L A7,SAVEA7
LEA.L DATA,A0
MOVEA.L A0,A3
JMP LAB3

   ;hier beginnt das Unterprogramm
LAB4:
MOVEA.L 0(A3),A4            ;lade 1. Parameter
MOVE.L (A4)+,D0
MOVE.L D0,16(A3)            ;speichere ihn nach A
      ;dieses A ist die lokale Variable in EX. Die
      ;ersten 16 Bytes sind u. a. von den beiden Para-
      ;metern belegt. Das Basisregister A3 wurde an
      ;der Aufrufstelle auf einen freien Speicher-
      ;bereich gelegt
MOVEA.L 4(A3),A4            ;lade 2. Parameter
MOVE.L (A4)+,D0
MOVEA.L 0(A3),A4            ;speichere auf 1. Parameter
MOVE.L D0, (A4)+

MOVE.L 16(A3),D0            ;hole Wert aus lokalem A
MOVEA.L 4 (A3),A4           ;speichere auf 2. Parameter
MOVE.L D0,(A4)+

MOVE.L 4*2(A3),RF+4*1       ;restauriere das Basisregister
MOVEA.L 4*2(A3), A1         ;auf den Wert beim Aufruf
RTS

   ;hier beginnt das Hauptprogramm
LAB3:
LEA.L 4(A3),A4              ;übergib 2. Parameter
MOVE.L A4,12(A3)
LEA.L 0(A3),A4              ;übergib 1. Parameter
MOVE.L A4,8(A3)
MOVE.L A3,–(A7)

MOVE.L A1,16(A3)            ;übergib bisheriges Basis-
                            ;register an EX
LEA.L 8(A3),A3              ;setze neues Basisregister
MOVEA.L A3,A1
MOVE.L A3,RF+4*1
JSR LAB4                    ;Aufruf von Ex
MOVEA.L (A7)+,A3

BRK: MOVEA.L SAVEA7,A7
MOVEM.L (A7)+,D0–D7/A0–A6
RTS
   :
   :
```

Bild 6. So verarbeitet der Compiler Unterprogramme mit lokalen Variablen

Michael Peth

Druckerumschaltung durch Type-Befehl

Bevor man sich die Arbeit macht, immer wieder die richtige Einstellung der Dip-Schalter am Drucker herauszufinden, lohnt es sich, einige Dateien auf einer Diskette anzulegen, die die Einstellung der gewünschten Schriftart sehr viel einfacher übernehmen. In eine Batch-Datei eingebunden, wird der Drucker automatisch richtig eingestellt.

Bei einem Computer mit angeschlossenem Drucker ergibt sich häufig das Problem, den Drucker auf einen bestimmten Zeichensatz, eine bestimmte Papierlänge oder ähnliches einzustellen. In vielen Fällen kann die korrekte Druckereinstellung bereits durch die DIP-Schalter vorgenommen werden. Einige Geräte erlauben sogar eine direkte Einstellung über ein Bedienungsfeld am Drucker. Bei allen anderen Geräten wird die richtige Einstellung im allgemeinen durch ESC-Sequenzen vorgenommen.

Mit Type den Drucker im Griff

Bei den meisten Heimcomputern, wie zum Beispiel C64 oder TI99-4A, kann die Übermittlung einer ESC-Sequenz zum Drucker durch selbstladende Basic-Programme vorgenommen werden. Beim IBM-PC und insbesondere bei kompatiblen Rechnern müßte jedoch erst das Basic mit dem Programm geladen werden. Nach der Übermittlung der Druckereinstellung befindet man sich dann im Basic, obwohl man vielleicht nur einige Diskettenverzeichnisse ausdrucken wollte. Die direkte Übermittlung einer ESC-Sequenz über die Tastatur scheitert im allgemeinen auch, da dies im Betriebssystem nicht vorgesehen ist. Wünschenswert wäre eine möglichst kurze Datei, welche eine Druckerumschaltung bewirkt und in der AUTOEXEC.BAT-Datei selbständig abgearbeitet wird. Das Erzeugen dieser Datei stößt jedoch auf Schwierigkeiten, da insbesondere der Zeichencode von ESC-CHR$(27) nicht so ohne weiteres dargestellt wird. Entweder probiert man mit einem Textverarbeitungsprogramm aus, ob der Druck auf die ESC-Taste einen nach links gerichteten Pfeil bewirkt, oder man muß sich des kurzen Basic-Programmes (*Bild*) bedienen, welches die benötigte Datei erzeugt. Das gespeicherte CHR$(27) ist, wenn man sich die Datei mittels „TYPE" ansieht, übrigens nicht sichtbar, bewirkt aber die gewünschte Umschaltung.

Alles weitere ist nach der Erzeugung der Dateien mit den ESC-Sequenzen recht einfach. Zur Übermittlung an den Drukker gibt man:

TYPE BREIT.ESC>PRN

in, und die BREIT.ESC-Datei wird dem Drucker übermittelt. Die Abarbeitung kann natürlich auch durch AUTOEXEC-.BAT-Datei, oder durch eine andere Batch-Datei erfolgen. Geeignet wäre zum Beispiel als Batch-Datei:

DRUCK.BAT MIT DEM Inhalt:
TYPE %1.ESC>prn

gibt man „DRUCK BREIT" ein, ist der Drucker auf Breitschrift eingestellt. „DRUCK NORMAL" bewirkt wieder normale Schriftbreite.

Im konkreten Fall wurde ein Mannesmann Tally Tintenstrahldrucker (MT90) auf eine Zeilenzahl von 72 (entspricht handelsüblichem Druckerpapier) gegenüber 66 Zeilen (US-Format) eingestellt.

Dieser Drucker erlaubt keine Voreinstellung mittels DIP-Schalter. Die ESC-Sequenz lautet in diesem speziellen Fall

CHR$(27);CHR$(67);CHR$(72).

Dieses Basic-Programm erzeugt zwei Dateien zur Druckerumschaltung

```
10 '
20 'Hier wird eine Datei für die Umschaltung auf Breitschrift erzeugt
30 '
40 OPEN "breit.esc" FOR OUTPUT AS #1
50 PRINT #1,CHR$(27);CHR$(87);CHR$(1) 'ESC-Sequenz für Breitschrift ein
60 CLOSE
70 '
80 'Hier wird die zweite Datei für die Umschaltung auf normale Breite erzeugt
90 '
100 OPEN "normal.esc" FOR OUTPUT AS #1
110 PRINT #1,CHR$(27);"W0" '            ESC-Sequenz für Breitschrift aus
120 CLOSE
```

Herbert Demmel

Cross-Assembler und EPROM

Hex-Dumps für den mc-EPROMer konvertieren

Für den IBM-PC und Kompatible stehen preisgünstige Hardware [1] und Software [2] zum Programmieren von EPROMs zur Verfügung. Dieser Artikel beschreibt, wie die von professionellen Cross-Assemblern erzeugten Hex-Dumps für das mc-EPROM-Programmiergerät verwendet werden.

In *Bild 1* ist ein Hex-Dump zu sehen, der von einem 65er-Cross-Assembler unter MS-DOS erzeugt wurde. Jede Zeile beginnt mit einem Doppelpunkt. Darauf folgt eine zweistellige Hexadezimalzahl, welche die Anzahl der genutzten Datenbytes angibt. Daneben steht vierstellig die Hex-Adresse des ersten Datenbytes in dieser Zeile. Programmiergeräte mit eingebauter Intelligenz benutzen diese Zahl, um die Datenbytes an der richtigen Stelle im EPROM zu plazieren.

Anschließend steht $00 als Trennzeichen zwischen der Adresse und den Datenbytes. Die beiden letzten Zeichen stellen die Prüfsumme dar, ebenfalls in Hexadezimaldarstellung. Von diesen Angaben sind nur die Datenbytes für den mc-Programmer brauchbar, die restlichen Informationen stören nur. Zum Berechnen der Prüfsumme werden alle in einer Zeile vorangegangenen Bytes addiert (einschließlich Adresse und Anzahl der Daten). Von diesem Ergebnis wird das 2er-Komplement gebildet und das niederwertige Byte als Prüfsumme verwendet.

Was das Programm kann ...

Das Programm (*Bild 2*) liest von der Diskette ein Hex-Dump, entfernt die für den mc-Programmer überflüssigen Angaben und schreibt den Rest zurück auf die Diskette. Dabei wird auch kontrolliert, ob die Prüfsumme stimmt. Sobald ein Fehler gefunden ist, wird die Konvertierung mit der entsprechenden Fehlermeldung abgebrochen. Die ausgegebene Fehlermeldung enthält die Adresse, an der der Fehler aufgetreten ist und z.B: den Soll- und Istwert der Prüfsumme.

Falls im Hex-Dump nicht alle Bytes mit Daten belegt sind, werden sie mit $FF aufgefüllt. Außerdem sorgt das Programm dafür, daß die Nutzdaten als

```
:10010000690565257525 6D55447D554479554461CE
:1001100025712572252905252535252D55443D5563
:1001200044395544212531253225 0A06251625 0E48
:1001300055441E55440F25FD1F25FD2F25FD3F2548
:10014000FD4F25FD5F25FD6F25FD7F25FD8F25FDDD
:100150009F25FDAF25FDBF25FDCF25FDDF25FDEF4B
:1001F000B95544A125B125B225A205A625B625AE3F
:100200005544BE5544A005A425B425AC5544BC5561
:10021000444A462556254E55445E5544EA0905058F
:10022000251525 0D55441D554419554401251125 05
:10023000122548 08DA5A6828FA7A0725172527254B
:10024000372547255725672577252A262536252E44
:1002E000F07FF07FF0802C050000010203AD035089
:0702F000070450040506 0796
:0000000000
```

Bild 1. So sieht eine Datei aus, die von einem Cross-Assembler erzeugt wird.

```
program c1;
const version = 'Version 1.0';
type f = string[2];
     f14 = string[14];
     f30 = string[30];
var kette : f;
    blockzahl, zaehler, bytezeiger, x: integer;
    source : text;
    target : file;
    offset : real;
    zelle : array[0..32767] of byte;
    sourcename,targetname : f14;
    abbruch, lastbyte : boolean;


function asc_dez(ziffern:f) : byte;
var z1,z2 : char;
    sub1,sub2 : integer;
begin
z1 := copy(ziffern,1,1);
z2 := copy(ziffern,2,1);
if ord(upcase(z1)) < 58 then sub1 := 48 else sub1 := 55;
if ord(upcase(z2)) > 63 then sub2 := 55 else sub2 := 48;
asc_dez := 16 * (ord(upcase(z1)) - sub1) + ord(upcase(z2)) - sub2
end;


procedure get_source_name;
begin
if paramstr(1) <> ''
   then sourcename := paramstr(1)
   else begin
        writeln;
        writeln('Bitte geben Sie die Quelldatei ein !');
        writeln;
        read(sourcename);
        end;
writeln;
writeln;
writeln('Quelldatei    : ',sourcename);
writeln;
end;
```

Bild 2. Mit diesem Programm werden Cross-Assembler-Dateien für den mc-EPROMer konvertiert

```
procedure get_target_name;
begin
targetname := sourcename;
writeln;
delete(targetname,length(sourcename)-2,3);
targetname := targetname + 'EPR';
writeln('Zieldatei     : ',targetname);
end;

procedure error(ursache : f30);
begin
writeln(chr(7));
writeln(ursache);
writeln('Konvertierung wird abgebrochen.');
abbruch := true;
writeln;
end;

procedure get_offset;
type fz = string[2];
var doppelpunkt : char;
    byte_je_zeile, adresse_hi,
    adresse_lo, checksum : fz;
    zweihoch8 : real;
begin
read(source,doppelpunkt,byte_je_zeile,adresse_hi,adresse_lo);
zweihoch8 := 256;
offset := zweihoch8 * asc_dez(adresse_hi) + asc_dez(adresse_lo);
writeln('Erste Adresse : $',adresse_hi,adresse_lo);
reset(source);
end;

procedure zeile;
type fz = string[2];
var doppelpunkt : char;
    byte_je_zeile, datum, adresse_hi,
    adresse_lo, trennzeichen, checksum : fz;
    bjz_dez, scratch, checksum_dez : byte;
    checksum_ist, i : integer;
    adresse_dez, zweihoch8 : real;

begin
checksum_ist := 0;
zweihoch8 := 256;
read(source,doppelpunkt);
read(source,byte_je_zeile,adresse_hi,adresse_lo,trennzeichen);
bjz_dez := asc_dez(byte_je_zeile);
checksum_ist := bjz_dez + asc_dez(adresse_hi) + asc_dez(adresse_lo);
adresse_dez := zweihoch8 * asc_dez(adresse_hi);
adresse_dez := adresse_dez + asc_dez(adresse_lo);
if bjz_dez = 0 then lastbyte := true;
if adresse_dez - offset - 1 > maxint - bjz_dez
   then error('Mehr als 32 kBytes');
if (bytezeiger > adresse_dez - offset) and (not lastbyte)
   then error('Falsche Reihenfolge der Zeilen');
if (not lastbyte) and (not abbruch) then
   begin
   if bytezeiger < adresse_dez-offset then
                repeat zelle[bytezeiger] := 255;
                       bytezeiger := bytezeiger + 1;
                until bytezeiger = round(adresse_dez-offset);
   for i := 0 to bjz_dez-1 do
       begin
       read(source,datum);
       scratch := asc_dez(datum);
       checksum_ist := checksum_ist + scratch;
       zelle[bytezeiger+i] := scratch;
       end;
   if bytezeiger < maxint then bytezeiger := bytezeiger + bjz_dez;
   read(source,checksum);
   checksum_dez := asc_dez(checksum);
   checksum_ist := checksum_ist mod 256;
   if  (checksum_ist + checksum_dez) mod 256 <> 0
       then begin
            writeln(chr(7));
            write('checksum error nach Adresse : $');
            write(adresse_hi,adresse_lo,checksum_ist:6);
            writeln('=?=',checksum_dez);
            writeln('Konvertierung wird nicht abgebrochen');
            end;
   readln(source);
   end;
end;


procedure ausgabe_crt;
begin
zaehler := 0;
for x := 0 to bytezeiger-1 do
    begin
    write(zelle[x]:4);
    zaehler := zaehler+1;
    if zaehler = 16 then begin
       zaehler := 0;
       writeln;
       end;
    end;
end;

(* hauptprogramm *)
begin
writeln;
write('hdCONV ',version);
get_source_name;
assign(source,sourcename);
reset(source);
get_offset;
bytezeiger := 0;
lastbyte := false;
abbruch := false;
repeat zeile
       until lastbyte or abbruch;
close(source);
if not abbruch
then begin
     (*ausgabe_crt;*)
     get_target_name;
     assign(target,targetname);
     writeln;
     blockzahl := round((bytezeiger)/128 + 0.5);
     writeln('Es werden ',blockzahl:0,' Blocks geschrieben');
     rewrite(target);
     blockwrite(target,zelle,trunc(blockzahl));
     close(target);
     end;
end.
```

„nicht typisierte Datei" für die Programmiersoftware aus [2] zur Verfügung stehen.

... und was es nicht kann.

Mit diesem Programm können keine Hex-Dumps verarbeitet werden, bei denen zwischen der niedrigsten und der höchsten Adresse mehr als 32 KByte liegen. Diese Einschränkung ist an den mc-Programmer angelehnt, der vom 2716 bis zum 27256 programmieren kann. Außerdem ist es nicht möglich, Hex-Dumps zu konvertieren, bei denen nicht alle Adressen in aufsteigender Reihenfolge vorliegen. In beiden Fällen würde die Konvertierung mit einer entsprechenden Fehlermeldung abgebrochen.

Nach dem Assemblerlauf steht z.B. die zu konvertierende Datei „TEST.HEX" auf Laufwerk B:. Das Konvertierprogramm wird mit „HDCONV B:TEST.HEX" aufgerufen. Der Name und das Laufwerk der Zieldatei werden aus der Angabe der Quelldatei abgeleitet. Die konvertierte Datei erhält automatisch die Extension ".EPR". Die Originaldatei bleibt natürlich unverändert erhalten.

Nach der Konvertierung steht in unserem Beispiel zusätzlich die Datei TEST.EPR auf Laufwerk B:. Nun kann die Programmiersoftware mit „EPROM" aufgerufen und die neue Datei geladen werden:

„LOAD B:TEST.EPR 0000" [2]

```
HDCONV B:test.hex

hdCONV Version 1.0

Quelldatei : B:test.hex

Erste Adresse : $0100

Zieldatei : B:test.EPR

Es werden 4 Blocks geschrieben
```

Bild 3. Die Benutzung des Programms ist einfach

In *Bild 3* ist ein kompletter Konvertiervorgang wiedergegeben.

Literatur:

[1] *Frank, Joachim*: Davor ist kein EPROM sicher, mc 8/1984, Seite 78

[2] *Schmidt, Hans-Jürgen*: EPROMs in MS-DOS programmiert, mc 3/1986, Seite 90

MS-DOS-Funktionstasten umbelegen

Mit einem kleinen Turbo-Pascal-Programm läßt sich die Belegung der Funktionstasten ändern. Das Programm läuft auf dem Siemens PC-D und dem Tandy 2000. Nach dem Compilieren wird das Programm als COM-Datei auf Diskette oder Festplatte abgelegt (Dateiname: F.COM). Die Systemdatei CONFIG.SYS muß den Befehl „DEVICE=ANSI.SYS" enthalten. Zum Starten des Programms muß F, gefolgt von der Funktionstastennummer und dem gewünschten Text, eingegeben werden. Steht hinter dem Text ein Doppelkreuz, so wird dieser mit einer CRLF-Sequenz abgeschlossen. Im Programmkopf zeigen zwei Beispiele, wie das Programm bedient wird.

Manfred Weis

```
program f;
   {*****************************************************************
    *              Programmierung von Funktionstasten               *
    *    Aufruf von MSDOS : f <Tastennummer> <Text> (Eingabetaste)  *
    *    Beispiele        : f 1 Turbo# (Eingabetaste)               *
    *                                   ^ bewirkt CRLF              *
    *                       : f 12 Dies ist eine Eingabezeile       *
    *****************************************************************}
var      tastennr,fehlercode,index,laenge :integer;
         zeichen                          :char;
         zeile                            :string[80];
begin
  val(paramSTR(1),tastennr,fehlercode);  { Einlesen der Tastennummer    }
  zeile:=paramSTR(2);                    { Einlesen der Textzeile       }
  for index:=3 to 40 do                  { Einen String bilden          }
      begin
        if length(paramSTR(index)) > 0 then
           zeile:= zeile+' '+paramSTR(index);
      end;
  if fehlercode = 0 then            { ist Tastennummer rein numerisch ? }
     begin write(#27#91#48#59);           { ESC [ 0 ; }
       case tastennr of                { Ausgabe der Funktionstastennummer }
              1    :    write(#53#57);       {  59  }
              2    :    write(#54#48);       {  60  }
              3    :    write(#54#49);       {  61  }
              4    :    write(#54#50);       {  62  }
              5    :    write(#54#51);       {  63  }
              6    :    write(#54#52);       {  64  }
              7    :    write(#54#53);       {  65  }
              8    :    write(#54#54);       {  66  }
              9    :    write(#54#55);       {  67  }
             10    :    write(#54#56);       {  68  }
             11    :    write(#49#54#48);    { 160  }
             12    :    write(#49#54#49);    { 161  }
             13    :    write(#56#52);       {  84  }
             14    :    write(#56#53);       {  85  }
             15    :    write(#56#54);       {  86  }
             16    :    write(#56#55);       {  87  }
             17    :    write(#56#56);       {  88  }
             18    :    write(#56#57);       {  89  }
             19    :    write(#57#48);       {  90  }
             20    :    write(#57#49);       {  91  }
             21    :    write(#57#50);       {  92  }
             22    :    write(#57#51);       {  93  }
             23    :    write(#49#54#50);    { 162  }
             24    :    write(#49#54#51);    { 163  }
       end;
       write(#59#34); { ; " }
       laenge:=length(zeile);
       zeichen:=copy(zeile,laenge,1);
       if zeichen <> '#' then
          begin for index:=1 to laenge do { Textstring ohne CR ablegen }
                begin zeichen:=copy(zeile,index,1);
                      write(chr(ord(zeichen)));
                end;
          write(#34#112); { " p }
          end
          else                                { '#': am Ende CRLF ausgeben }
          begin for index:=1 to laenge-1 do
                begin zeichen:=copy(zeile,index,1);
                      write(chr(ord(zeichen)));
                end;
          write(#34#59#49#51#112); { " ; 1 3 p }
          end;
     end;
end.
```

So lassen sich beim Siemens PC-D und beim Tandy 2000 die Funktionstasten umbelegen

Gerhard Röhner

Funktions-interpreter in Turbo-Pascal

Die Programmiersprache MBASIC bietet reichlich Komfort, wenn es darum geht, Funktionen zu verarbeiten. Ohne großen Aufwand kann man eine Funktion im Klartext einlesen und mit Chain und Merge dem Programm einverleiben. Der Basic-Interpreter kann anschließend den Funktionsterm auswerten. Welche Mühen hat man dagegen in Pascal, Gleichartiges zu realisieren. Da können eigentlich nur zum Compilier-Zeitpunkt bekannte Funktionen benutzt werden. Für jede neue Funktion muß man das Programm neu übersetzen. Hier hilft ein eigener Funktionsinterpreter.

Die erste Aufgabe eines Interpreters besteht darin, den als Zeichenkette eingegebenen Funktionsterm zu analysieren. Analysieren heißt, den Term in syntaktische Einzelteile wie Operatoren, Zahlen, Funktionen und Vorzeichen zu zerlegen und die Hierarchie der Teile zu ermitteln. Die dieser Aufgabe angemessene Datenstruktur ist der Baum. So kann beispielsweise der Term
−x*sin(3^x−2)/ln(3+x^2)
als Baum wie in *Bild 1* dargestellt werden. In den Blättern des Baumes stehen Konstanten oder die Variable x. Die restlichen Knoten des Baumes stellen Operatoren oder Funktionen dar. Das „−"-

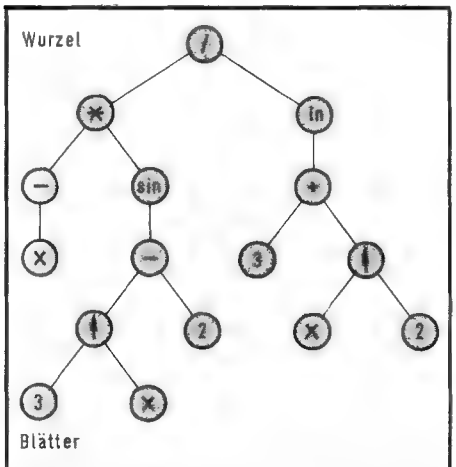


Bild 1. Der Term −x*sin(3^x−2)/ln(3+x^2) als Baum dargestellt

Zeichen kann als einstelliger Operator in der Bedeutung eines Minuszeichens auftreten oder als zweistelliger Operator zur Kennzeichnung einer Differenz. Die restlichen Operatoren (+,*,/,^) sind stets zweistellig und haben daher zwei Unterbäume. Funktionen haben immer einen Unterbaum, der das Argument der Funktion bildet. Klammerterme können als Argumente der Identitätsfunktion id (x) = x aufgefaßt werden. Diese Interpretation der Klammern vereinfacht den gewählten Analyseprozeß.
Die unterschiedliche Belegung der Knoten des Baumes durch Operatoren, Funktionen, Vorzeichen, Zahlen und Variablen spiegelt sich in der Datenstruktur wieder (*Bild 2*). Die Knoten werden als variante Records deklariert. Beim Bau des Baumes müssen die üblichen Rechenregeln
− Punkt- vor Strichrechnung,
− Klammern zuerst,
− von links nach rechts
beachtet werden. Sie lassen sich in einer Hierarchie der Rechenoperationen zusammenfassen:
1. Klammer und Funktionen
2. Potenz ^
3. Vorzeichen +,−
4. Punktrechnung *,/
5. Strichrechnung +,−
Höherwertige Operationen befinden sich im Baum unten bei den Blättern, denn die Berechnung eines Baumes beginnt bei den Blättern. Die Wurzel des Baumes wird hingegen von der niedrigwertigsten Operation im Term bestimmt.

Eingaberegeln beachten

Für einen strukturierten Analyse-Algorithmus ist es erforderlich, den Begriff des Funktionsterms präzise zu definieren. Zunächst einmal müssen alle Rechenzeichen geschrieben werden, der Malpunkt darf also nicht weggelassen werden. Grundsätzlich dürfen nie zwei Operatoren direkt nebeneinander stehen. Die Ausnahme von dieser Regel bilden Vorzeichen für den Exponenten einer Potenz. Ansonsten sind beliebig komplizierte Ausdrücke zugelassen.
Diese Erklärungen reichen für einen Anwender aus, für den Programmentwurf nicht. Hier verwendet man Syntaxdiagramme, welche die Struktur eines gültigen Ausdrucks am anschaulichsten darstellen. Die Syntaxdiagramme in *Bild 3* definieren das, was hier unter einem Term verstanden werden soll.
Elegante Analysealgorithmen untersuchen einen Term zeichenweise, von links nach rechts durchlaufend. Dem entspricht im Baum ein Weg vom linken Blatt über die Wurzel zum rechten Blatt. Es wird also keineswegs mit der Baumwurzel begonnen. Aus diesem Grund sind solche Algorithmen auch relativ schwierig zu verstehen.
Für das Verständnis ist es günstiger, den Term an der Wurzel zu packen und sich zu den Blättern vorzuarbeiten. Der Analysealgorithmus ist dann zwar nicht mehr so effektiv, aber das spielt hier keine Rolle, weil das Analyseergebnis (der Baum) im Gegensatz zur Arbeitsweise von Basic-Interpretern nicht weggeworfen wird.
In Bild 2 finden Sie den Algorithmus „FunktionsAnalyse". Er arbeitet sich gemäß der Rechenregel „von links nach rechts" in umgekehrter Richtung durch den Funktionsterm, sucht unter Beachtung der Hierarchie der Rechenregeln Operatoren im Term und hängt die Operanden als Unterbaum in die Knoten ein. Die rekursiven Prozeduraufrufe enden in den Blättern, bei den Zahlen und Variablen.
Wenn Sie es gerne eleganter hätten, einen Algorithmus, der den Syntaxdiagrammen direkt entspricht, dann schauen Sie sich auf der Turbo-Pascal-System-Diskette das Demonstrationsprogramm MicroCalc von Borland an. Es ist zwar in der Version 1.0 nicht korrekt (z. B.: „sin(3" ohne schließende Klammer wird akzeptiert, −2^3 kann nicht berechnet werden, weil es fälschlicherweise als (−2)^3 interpretiert wird), gibt aber Hin-

```
{*********************************************************************}
{* Quellcode FUNKTION.INC  zur interaktiven Eingabe von Funktionen *}
{* von G.Roehner, Darmstadt                          ,den 5.10.86 *}
{*********************************************************************}

Type Arttyp    = (Vorzeichen,Operator,Funktion,Zahl,VarX);
     String128 = String[128];
     OpTyp     = Set Of Char;
     Ptr       = ^Knotentyp;
     Knotentyp = RECORD CASE Art:Arttyp Of
                   Vorzeichen : (von:Ptr);
                   Operator   : (Op:Char; links,rechts:Ptr);
                   Funktion   : (Fn:String[6]; next:Ptr);
                   Zahl       : (Za:REAL);
                   VarX       : ();
                 END;
{---------------------------------------------------------------------}

Function SucheOperator(Ops:OpTyp; Var k:Integer; Term:String128):Boolean;
    Var Klammer:Integer; Gefunden:Boolean;
Begin
  Gefunden:=FALSE; Klammer:=0; k:=Length(Term)+1;
  Repeat
    k:=k-1;
    Case Term[k] Of
      '( : Klammer:=Pred(Klammer);
      ')': Klammer:=Succ(Klammer);
    else if (Klammer=0) AND (Term[K] IN Ops) AND (k>1)
            AND (Not (Term[k-1] IN ['E','^']))
         then Gefunden:=TRUE
    end;
  until Gefunden OR (k=1);
  SucheOperator:= Gefunden;
End;

Function SuchePlusMinus(Var k:Integer; Term:String128):Boolean;
Begin
  SuchePlusMinus:=SucheOperator(['+','-'], K, Term);
End;

Function SucheMalDurch(Var k:Integer; Term:String128):Boolean;
Begin
  SucheMalDurch:=SucheOperator(['*','/'], K, Term);
End;

Function SucheVorzeichen(Term:String128):Boolean;
Begin
  SucheVorzeichen:=(Term[1] IN ['+','-']);
end;

Function SuchePotenz(Var k:Integer; Term:String128):Boolean;
Begin
  SuchePotenz:=SucheOperator(['^'], K, Term);
end;

Function SucheFunktion(Var k:Integer; Term:String128):Boolean;
  Var F:String128;
Begin
  SucheFunktion:=FALSE; k:=POS('(',Term);
  If k>0 Then Begin
    F:=Copy(Term,1,k-1);
    If ( (F='ABS') OR (F='ARCTAN ) OR (F='COS') OR (F='EXP') OR
         (F='FRAC') OR (F='INT') OR (F='LN') OR (F='SIN') OR
         (F='SQR') OR (F='SQRT') OR (F='TAN') OR (F=''))
        AND (Term[Length(Term)]=')')
    Then SucheFunktion:=TRUE
    End;
End;

Function SucheZahl(Var Wert:REAL; Term:String128):Boolean;
  Var Code:Integer;
Begin
  Val(Term,Wert,Code);
  SucheZahl:=(Code=0);
End;

Function SucheX(Term:String128):Boolean;
Begin
  SucheX:=(Term='X');
End;

{---------------------------------------------------------------------}

Function FunktionsAnalyse(Term:String128):Ptr;

  Var TermOk:Boolean; Fkt:Ptr;

  Function UpDate(Term:String128):String128;
    Var i:Integer; Ch:Char; UpTerm:String128;
  Begin
    UpTerm:='';
    For i:=1 To Length(Term) Do Begin
      Ch:=UpCase(Term[i]);
      If Ch<>' ' Then UpTerm:=UpTerm+Ch;
      End;
    UpDate:=UpTerm;
  End;

  {$A- rekursive Prozedur }

  Procedure BaueBaum(VAR Knoten:Ptr; Term:String128);
    Var Wert:REAL; k:Integer; Vz:Char;
  Begin
   If TermOk and (Length(Term)>0) Then
    If SuchePlusMinus(k,Term) Then Begin
      New(Knoten);
      Knoten^.Art:=Operator; Knoten^.Op :=Term[k];
      BaueBaum(Knoten^.links ,Copy(Term,1,k-1));
      BaueBaum(Knoten^.rechts,Copy(Term,k+1,Length(Term)-k));
      End
    Else If SucheMalDurch(k,Term) Then Begin
      New(Knoten);
      Knoten^.Art:=Operator; Knoten^.Op :=Term[k];
      BaueBaum(Knoten^.links ,Copy(Term,1,k-1));
      BaueBaum(Knoten^.rechts,Copy(Term,k+1,Length(Term)-k));
      End
    Else If SucheVorzeichen(Term) Then Begin
      Vz:=Term[1]; Delete(Term,1,1);
      Case Vz of
        '+': BaueBaum(Knoten, Term);
        '-': Begin
               New(Knoten); Knoten^.Art:=Vorzeichen;
               BaueBaum(Knoten^.von, Term);
             end;
      end
    Else If SuchePotenz(k,Term) Then Begin
```

Bild 2. Pascal-Prozeduren zur Analyse von Funktionen, die während des Programmablaufs eingegeben werden

```
      New(Knoten);
      Knoten^.Art:=Operator; Knoten^.Op :=Term[k];
      BaueBaum(Knoten^.links ,Copy(Term,1,k-1));
      BaueBaum(Knoten^.rechts,Copy(Term,k+1,Length(Term)-k));
      End
    Else If SucheFunktion(k,Term) Then Begin
      New(Knoten);
      Knoten^.Art:=Funktion; Knoten^.Fn :=Copy(Term,1,k-1);
      BaueBaum(Knoten^.next,Copy(Term,k+1,Length(Term)-1-k));
      End
    Else If SucheZahl(Wert,Term) Then Begin
      New(Knoten);
      Knoten^.Art:=Zahl; Knoten^.Za :=Wert;
      End
    Else If SucheX(Term) Then Begin
      New(Knoten);
      Knoten^.Art:=VarX;
      End
     Else TermOk:=False
   Else TermOk:=False;
   End;
   {$A+}

Begin { FunktionsAnalyse }
  TermOk:=True; Term:=UpDate(Term);
  BaueBaum(Fkt,Term);
  If Not TermOk
    Then FunktionsAnalyse:=NIL
    Else FunktionsAnalyse:=Fkt;
End;

{-------------------------------------------------------------------------}

Function FunktionsBerechnung(Fkt:Ptr; X:Real):Real;

  CONST MaxReal=1.0E+30; MinReal=1.0E-30;
  Var   WertOk:Boolean;

  {$A- Rekursive Funktion }

  Function Wert(Zeiger:Ptr):Real;
    Var Test:Real;
  Begin
    If WertOk Then
      With Zeiger^ Do
        Case Art Of
          Vorzeichen: Wert:=-Wert(von);
          Operator:Case Op Of
                     '+':Wert:=Wert(links)+Wert(rechts);
                     '-':Wert:=Wert(links)-Wert(rechts);
                     '*':Wert:=Wert(links)*Wert(rechts);
                     '/':Begin
                           Test:=Wert(rechts);
                           If ABS(Test)>MinReal
                             Then Wert:=Wert(links)/Test
                             Else WertOk:=False;
                         End;
                     '^':Begin
                           Test:=Wert(links);
                           If Test>MinReal
                             Then Wert:=EXP(Wert(rechts)*LN(Test))
                             Else WertOk:=False;
                         End;
                   End;
```

```
          Funktion:Begin
                     If Fn=''       Then Wert:=Wert(next);      { Klammern }
                     If Fn='ABS'    Then Wert:=ABS(Wert(next));
                     If Fn='ARCTAN' Then Wert:=ARCTAN(Wert(next));
                     If Fn='COS'    Then Wert:=COS(Wert(next));
                     If Fn='EXP'    Then Wert:=EXP(Wert(next));
                     If Fn='FRAC'   Then Wert:=FRAC(Wert(next));
                     If Fn='INT'    Then Wert:=INT(Wert(next));
                     If Fn='LN'     Then Begin
                                           Test:=Wert(next);
                                           If Test>0
                                             Then Wert:=LN(Test)
                                             Else WertOk:=False;
                                         End;
                     If Fn='SIN'    Then Wert:=SIN(Wert(next));
                     If Fn='SQR'    Then Wert:=SQR(Wert(next));
                     If Fn='SQRT'   Then Begin
                                           Test:=Wert(next);
                                           If Test>0
                                             Then Wert:=SQRT(Test)
                                             Else WertOk:=False;
                                         End;
                     If Fn='TAN'    Then Wert:=SIN(Wert(next))/COS(Wert(next));
                   End;
          Zahl    :Wert:=Za;
          VarX    :Wert:=X;
          End;
    End;
    {$A+}

begin { FunktionsBerechnung }
  If Fkt<>NIL Then Begin
    WertOk:=TRUE; FunktionsBerechnung:=Wert(Fkt);
    If  Not WertOK Then FunktionsBerechnung:=MaxReal
    end
  else FunktionsBerechnung:=MaxReal;
end;

{-------------------------------------------------------------------------}

{$A- rekursive Prozedur }

Procedure ZeigeBaum(Fkt:Ptr; Tiefe:Integer);
Begin
  If Fkt<>NIL Then
   With Fkt^ DO
    Case Art Of
      Vorzeichen: Begin Writeln(' ':Tiefe,'-');
                        ZeigeBaum(von,Tiefe+3);
                  End;
      Operator:   Begin ZeigeBaum(rechts,Tiefe+3);
                        Writeln(' ':Tiefe,Op);
                        ZeigeBaum(links ,Tiefe+3)
                  End;
      Funktion:   Begin if fn='' then Writeln(' ':tiefe,'()')
                                 else Writeln(' ':Tiefe,Fn);
                        ZeigeBaum(next,Tiefe+3)
                  End;
      Zahl:       Writeln(' ':Tiefe,Za);
      VarX:       Writeln(' ':Tiefe,'X');
      End;
End;
{$A+}
```

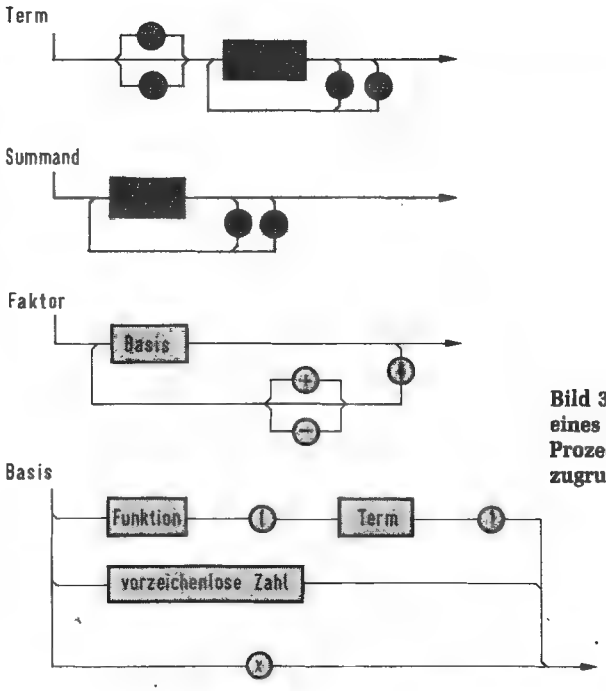


Bild 3. Diese Definition eines Terms liegt den Prozeduren von Bild 2 zugrunde

weise, wie die Syntaxdiagramme programmiert werden können. Da hier bei den Blättern begonnen wird, ist es etwas schwieriger, den Baum zu erstellen. Versuchen Sie es dennoch!

Term für beliebige x-Werte auswerten

Jetzt kommen wir zur zweiten Aufgabe unseres Interpreters. Der Term muß für beliebige x-Werte ausgewertet werden. Dies leistet der Algorithmus „Funktions-Berechnung" in Bild 2. Er enthält als Eingangsparameter den Zeiger auf die Funktion und den x-Wert und liefert als Ergebnis eine reelle Zahl, den Funktionswert. Die Berechnung geschieht rekursiv, wobei der Baum von der Wurzel zu den Blättern, in der Reihenfolge linker Teilbaum, Knoten, rechter Teilbaum, also „Inorder", durchlaufen wird. Typische, aber nicht alle Fehlersituationen werden abgefangen(Division durch Null, negative Basis bei Potenzen, Wurzeln mit negativem Radikanden). In solchen Fällen erhält man den Funktionswert: MaxReal. Die Lösung bietet sich bei Funktionsplottern an, weil MaxReal sicherlich außerhalb der Zeichenfläche liegt. Selbst beim Addieren (9E37 + 9E37) können Laufzeitfehler auftreten. Sie können aber alle Fehler durch geeignete Anweisungen abfangen. Leider verringert sich dadurch die Rechengeschwindigkeit. Finden Sie Ihren Kompromiß.

Für Lernende ist die Prozedur „ZeigeBaum" gedacht. Sie gibt den Baum eines Terms wie in *Bild 4* wieder. Das Demoprogramm (*Bild 5*) zeigt die Verwendung der Prozeduren in einem Anwenderprogramm.

```
-x*sin(3^x-2)/ln(3+x^2)

  LN
                2.0000000000E+00
          ^
             X
     +
          3.0000000000E+00
/
     SIN
                2.0000000000E+00
        -
             X
           ^
                   3.0000000000E+00
  *
     -
        X
```

Bild 4. So gibt die Prozedur „ZeigeBaum" den Baum eines Terms wieder

```
Program Demo;

  {$I B:FUNKTION.INC }

  Var Wahl : Char;
      X,Y  : Real;
      Fkt  : Ptr;
      Term : String128;

Begin
  Repeat
    ClrScr;
    GotoXY(3,2); Write('Interaktive Funktionseingabe');
    GotoXY(3,4); Write('<1> Term eingeben');
    GotoXY(3,5); Write('<2> Baum zeigen');
    GotoXY(3,6); Write('<3> Funktionswerte');
    GotoXY(3,7); Write('<4> Ende');
    Repeat
      GotoXY(7,9); Write('Ihre Wahl: ');
      Read(KBD,Wahl); Write(Wahl);
    Until Wahl In ['1'..'4']; GotoXY(7,9);
    Case Wahl Of
      '1':Begin
            Write('Funktionsterm: '); Readln(Term);
            Fkt:=FunktionsAnalyse(Term);
            If Fkt=NIL Then Begin
              GotoXY(7,11); Write('Fehler im Term');
              Readln End;
          End;
      '2':Begin
            GotoXY(7,11); Writeln(Term);Writeln;
            ZeigeBaum(Fkt,0); Readln;
          End;
      '3':Begin
            If Fkt<>NIL Then Begin
              Write('X = '); ClrEol;
              GotoXY(11,9); Readln(X);
              Y:=FunktionsBerechnung(Fkt,X);
              GotoXY(7,11); Writeln('Y = ',Y);
              End
            Else Begin
                   GotoXY(7,11);
                   Write('Keine Funktion da.');
                 End;
            Readln;
          End;
    End;
  Until Wahl='4';
End.
```

Bild 5. Dieses Programm demonstriert die Anwendung der Prozeduren aus Bild 2

Literatur:

[1] *Wirth, Nikolaus:* Algorithmen und Datenstrukturen, B. G. Teubner, Stuttgart, 1983.
[2] *Wirth, Nikolaus:* Compilerbau, B. G. Teubner, Stuttgart, 1986.

Raul Rojas

Move-Utility für MS-DOS

MS-DOS wird mit einem Unix-ähnlichen Befehl erweitert

MS-DOS steht mit seiner Leistungsfähigkeit ungefähr in der Mitte zwischen CP/M und Unix. MS-DOS und CP/M lassen sich vom Anwender leicht bedienen, Unix hat jedoch für Programmierer mächtigere Befehle als MS-DOS. Unser Autor zeigt, wie MS-DOS um den Move-Befehl erweitert wird.

Einige MS-DOS-Befehle haben den gleichen Namen wie CP/M-Befehle (DIR, TYPE, ERA). MS-DOS hat aber auch einiges von Unix übernommen. In beiden Betriebssystemen werden hierarchische Datei-Strukturen verwaltet, Ausgaben lassen sich umleiten und Pipes können verwendet werden. Etliche Befehle haben in Unix seltsame Namen. Sie lassen sich aus ihrer erweiterten Funktionalität gegenüber MS-DOS-Kommandos erklären. Soll in Unix eine Datei gelöscht werden, so heißt der Befehl rm (remove). In MS-DOS wird DEL (delete) benutzt. Delete kann vom Anwender leichter auswendig gelernt werden. Der Remove-Befehl kann jedoch mehr als der Delete-Befehl, da in der Unix-Dateistruktur eine Datei in mehreren Unterverzeichnissen eingetragen sein kann, ohne daß dafür mehrere Kopien der Datei existieren. Nur der Dateiname und ein Zeiger auf den Anfang der Datei werden in so vielen Unterverzeichnissen wie gewünscht gespeichert. Remove löscht eine dieser Eintragungen aus einem Unterverzeichnis, läßt aber die anderen unberührt. Nur wenn die letzte Eintragung gelöscht wird, geht die Datei endgültig verloren.

Ähnliches geschieht mit den Befehlen Rename (MS-DOS) und Move (Unix). Mit REN kann eine Datei in MS-DOS und mit MV in Unix umbenannt werden. „Move" kann aber mehr als „Rename", denn es ist möglich, eine Datei von einem Unterverzeichnis in ein anderes zu übertragen. Soll z. B. die Datei file1 nicht mehr in dem Unterverzeichnis /bin stehen, sondern in /sys ,dann erreicht man dies mit dem Befehl: mv /bin/file1 /sys/file1

Die Datei wird nicht kopiert – nur die Eintragungen in den Unterverzeichnissen werden geändert: in /bin wird die Eintragung und der Zeiger auf file1 gelöscht, in /sys wird der Name und der Zeiger auf file1 notiert.

In MS-DOS ist so etwas nicht ohne weiteres möglich. Um eine Datei von einem Unterverzeichnis in ein anderes zu verlegen, muß zuerst die Datei kopiert werden und danach die alte Datei gelöscht werden. Einzugeben ist:

```
copy \bin\file1 \sys\file1
del \bin\file1
```

Die Unix-Lösung ist eleganter. Die Dateiverlegung ist nicht nur schneller, sondern es werden auch andere Probleme umgangen. Wenn in MS-DOS bereits eine Diskette voll ist und der Platz nicht ausreicht, um eine zweite Kopie einer Datei zu machen, dann funktioniert die obige Kombination nicht ohne weiteres. Auf einer Festplatte tritt dieses Problem nur selten auf – von Bedeutung ist jedoch die Fragmentierung von Dateien auf der Festplatte.

In MS-DOS sind die Datenträger (Festplatten oder Disketten) in Cluster mit je vier bzw. zwei 512-Byte-Sektoren aufgeteilt. Um Daten abzuspeichern, werden immer die niedrigsten freien Cluster verwendet. In unserem Beispiel werden neue Cluster belegt, wenn die Datei kopiert wird. Sobald die alte Datei gelöscht wird, werden einige Cluster frei, die aber in den meisten Fällen eine Lücke auf der Diskette oder Festplatte hinterlassen. Diese Lücken werden von anderen neuen Dateien benutzt. Falls die Lücke zu klein ist, werden die Dateien auf dem Speichermedium fragmentiert. Ein Teil

```
;****************************************
;*  MOVE-UTILITY                        *
;*  Raul Rojas  Nov. 1986               *
;*                                      *
;* >MASM MOV                            *
;* >LINK MOV                            *
;* >EXE2BIN MOV MOV.COM                 *
;****************************************

CODE    SEGMENT
        ORG 100h                ;ORG-100 für COM-Datei
        ASSUME CS:CODE,DS:CODE;
start:  mov ax,cs               ;
        mov ds,ax               ;Data-Segment=Code-Segment
        mov bx,80h              ;Hier fängt die Argumentzeile an
        xor cx,cx               ;clear CX
        mov cl,byte ptr[bx]     ;CX zeigt auf das Ende der
        add cx,81h              ;Eingabezeile
skip:   inc bx                  ;nächstes Zeichen
        cmp cx,bx               ;überprüfe, ob noch Zeichen
        jb fehler               ;da sind
        cmp byte ptr [bx],20h   ;Blank?
        jz skip                 ;siehe, ob nächstes Zeichen
                                ;auch Blank ist
        mov dx,bx               ;DX zeigt auf den ersten Dateinamen
asciz:  inc bx                  ;nächstes Zeichen
        cmp cx,bx               ;überprüfe, ob noch Zeichen
        jb fehler               ;da sind
```

Bild 1. Das Assembler-Listing der Move-Utility

einer neuen Datei könnte in den gerade freigewordenen Cluster gespeichert werden, während der Rest in anderen Clustern, die physisch weit entfernt sein können, gespeichert wird. Das Resultat ist, daß der Lesekopf beim Lesen der Datei auf der Platte „herumschabt".

Mit dem hier vorgestellten Programm (*Bild 1*) kann der Move-Befehl auch unter MS-DOS verwendet werden. Microsoft hat dies bereits mit der DOS-Funktion 56H des Interrupts 21H vorbereitet. Diese Funktion arbeitet wie das mv-Kommando in Unix. Die Register DX und DI müssen nur auf die Dateinamen (den alten und den neuen) zeigen. Den Dateinamen kann ein Pfad vorangestellt werden.
Der größte Teil des Programms (*Bild 2*) besteht aus der Analyse der Eingabezeile. Ab Adresse 81H werden die Parameter der externen DOS-Befehle gespeichert. Um die Zeiger auf den Anfang der Dateinamen zu richten, müssen Leerzeichen übersprungen werden. Am Ende der beiden Dateinamen muß ein Null-Byte stehen, da die Funktion 56H zwei ASCIIZ-Strings erwartet. In 80H wird die Zahl der getippten Zeichen gespeichert. Damit wird kontrolliert, ob das „Scanning" bereits beendet ist.

Mit mov kann eine Datei wie in Unix verlegt werden:
mov \bin\file1 \sys\file1

```
62A5:0100  8C C8 8E D8 BB 80 00 33-C9 8A 0F 81 C1 81 00 43   .H.X;..3I...Á..C
62A5:0110  3B CB 72 3A 80 3F 20 74-F6 8B D3 43 3B CB 72 2E   ;Kr:.? tv.SC;Kr.
62A5:0120  80 3F 20 75 F6 C6 07 00-43 3B CB 72 21 80 3F 20   .? uvF..C;Kr!.?
62A5:0130  74 F6 1E 07 8B FB 43 3B-CB 72 13 80 3F 20 74 05   tv...{C;Kr..? t.
62A5:0140  80 3F 0D 75 F1 C6 07 00-B4 56 CD 21 73 07 BA 59   .?.uqF..4VM!s.:Y
62A5:0150  01 B4 09 CD 21 B4 00 CD-21 46 65 68 6C 65 72 20   .4.M!4.M!Fehler
62A5:0160  69 6E 20 4D 4F 56 0D 0A-24                        in MOV..$
```

Bild 2. Der Hex-Dump

Ist es notwendig, mehrere Dateien auf einmal zu verlegen, so hilft folgende Stapel-Datei:

```
for %%a in (%1) do mov %%a %2%%a
```

Wenn die Stapeldatei xmov.bat heißt und wir uns in dem Unterverzeichnis befinden, aus dem die Dateien verlegt werden sollen, dann muß das erste Argument (%1) ein Dateiname sein. Globale Dateinamenbezeichner wie ? und * dürfen benutzt werden. Das zweite Argument ist der neue Pfad. Es ist möglich, folgendes einzugeben:

```
xmov *.com \bin\
```

Damit werden alle COM-Dateien aus dem aktiven Unterverzeichnis nach \bin\ verlegt.

```
                cmp byte ptr [bx],20h     ;Blank?
                jnz asciz                 ;wenn nicht, hole nächstes Zeichen
                mov byte ptr [bx],0       ;Null am Ende des ersten Namens
skip2:          inc bx                    ;nächstes Zeichen
                cmp cx,bx                 ;überprüfe, ob noch Zeichen
                jb fehler                 ;da sind
                cmp byte ptr [bx],20h     ;Blank?
                jz skip2                  ;wenn ja, hole nächstes Zeichen
                push ds                   ;
                pop es                    ;ES zeigt jetzt auf CODE-Segment
                mov di,bx                 ;zweiten Pfad in DI
asciz2:         inc bx                    ;nächstes Zeichen
                cmp cx,bx                 ;überprüfe, ob noch Zeichen
                jb fehler                 ;da sind
                cmp byte ptr [bx],20h     ;Blank?
                jz fertig                 ;wenn ja, sind wir fast fertig
                cmp byte ptr [bx],0dh     ;Return?
                jnz asciz2                ;wenn nicht, hole nächstes Zeichen
fertig:         mov byte ptr [bx],0       ;Null am Ende
                mov ah,56h                ;Function 56H (Change Directory
                int 21h                   ;Entry)
                jnc done                  ;Kein Carry=alles in Ordnung
fehler:         mov dx,offset error       ;Falls Carry gesetzt ist
                mov ah,09h                ;gab es einen Fehler
                int 21h                   ;Zeigt den Fehler an
done: mov ah,0                            ;Programm
                int 21h                   ;beenden
;
error           db "Fehler in MOV",0dh,0ah,"$"
CODE            ENDS
END             start
```

LAN und Telekommunikation

Das Hauptanwendungsgebiet der lokalen Computernetzwerke (LAN) ist die Datenkommunikation, während Telekommunikationsanlagen, wie z. B. Nebenstellenanlagen, der Sprachkommunikation dienen. Mit ISDN (Integrated Services Digital Network) stehen den Anwendern zwei Übertragungskanäle mit je 64 KBit/s und ein Signalisierungskanal mit 16 KBit/s zur Verfügung. In Zukunft können Datenkommunikation, Breitband-Koppelnetze für Bildkommunikation in den Telekommunikationsanlagen integriert werden. Welche Unterschiede bestehen zwischen LANs und Telekommunikationsanlagen?

Die Architektur von LANs weicht von der Sternform der Leitungsnetze traditioneller Nebenstellenanlagen ab und baut auf Bus-, Baum oder Ringstrukturen auf. Ein weiteres wichtiges Unterscheidungsmerkmal ist die Steuerung der Zugriffe auf die jeweiligen Übertragungsmedien. Mit LANs werden Systeme mit verteilter Intelligenz realisiert. Ihre Steuerung erfolgt dezentral. Der damit verbundene Konflikt der Vergabe des gemeinsamen Übertragungsmediums wird durch Zugriffsprotokolle gelöst. Bei Telekommunikationsanlagen treten derartige Zugriffskonflikte nicht auf, da dort die Steuerung zentral erfolgt.

Die Bandbreite, der von LANs bereitgestellten Übertragungsmedien, liegt für Basisbandsysteme, die ihren Einsatzschwerpunkt in der Vernetzung von Bildschirmarbeitsplätzen auf Abteilungsebene haben, im Bereich 100 KBit/s...10 MBit/s. Infrastruktur-orientierte LANs (Backbone-LAN), mit denen die Verkabelung eines gesamten Firmengeländes möglich ist, können Übertragungsraten bis zu einigen 100 MBit/s erreichen. *Nach Unterlagen vom ZVEI*

Jürgen Schröder

Zeitanzeige in Fenstertechnik

Die Echtzeituhr des Siemens PC-D läßt sich einfach ansprechen

Wie im Siemens PC-D schlummert in vielen PCs eine batteriegepufferte Echtzeituhr. mc zeigt wie die laufende Zeit an beliebiger Stelle des Bildschirms angezeigt wird, ohne daß die Bedienung des PCs eingeschränkt wird.

Das Zeitanzeige-Programm läuft unverändert nur auf dem Siemens PC-D. Das Assembler-Programm (*Bild 1*) ist so dokumentiert, daß es an andere PCs, sofern sie über eine Echtzeituhr verfügen, angepaßt werden kann.
Alle 50 ms wird im PC der Hardwareinterrupt 8H ausgelöst. Die darauf folgende Interrupt-Service-Routine ruft den Interrupt 1CH auf. Der zugehörige Vektor zeigt auf eine IRET-Anweisung (Return from Interrupt = Rücksprung nach einem Interrupt), so daß der 1CH-Interrupt ohne Wirkung ist. Man kann diesen Vektor „umbiegen" und auf eine Routine zeigen lassen, um die aktuelle Zeit anzuzeigen. Diese Routine muß mit IRET abgeschlossen werden.
Da die Routine alle 50 ms aufgerufen wird, muß sie speicherresident sein und darf durch andere Programme nicht überschrieben werden.

Parameter übergeben die Anzeigeposition

Mit dem Befehl ZEIT Z,S kann man die Zeile (0...24) und die Spalte (0...72) der Zeitanzeige angeben. Beim Aufruf wird zunächst zum nichtresidenten Teil des Programms gesprungen und anhand des 1CH-Interrupts überprüft, ob die Anzeigeroutine schon resident ist. Falls die Anzeigeroutine bereits geladen ist, werden die Parameter, die in der Kommandozeile nach ZEIT stehen, überprüft. ZEIT ohne Parameter unterdrückt die Anzeige.
Der 1CH-Vektor wird beim ersten Aufruf durch die DOS-Funktion 25H auf die residente Routine umgebogen, die Anzeige aktiviert und das Programm im Speicher durch die Funktion 31H geschützt. Da die Anzeige nur jede Sekunde aktualisiert (20 x 50 ms) werden soll, wird ein Zähler alle 50 ms inkrementiert. Ist eine Sekunde vergangen, so muß noch überprüft werden, ob sich der PC im Grafikmodus befindet (BIOS-Datensegment, Offset 49H). Eine Anzeige im Grafik-Modus kann beim Umschalten in den normalen Anzeigemodus einen Systemabsturz verursachen. Der letzte Teil des Programms rettet die jeweilige Cursorposition, liest den Uhrenport (beim Siemens PC-D F884H, F882H, F880H) und schreibt die Stunden, Minuten und Sekunden auf den Bildschirm. Das Schreiben erfolgt im DISPLAY-Unterprogramm mit INT 29H. Mit IRET erfolgt der Rücksprung.
Bild 2 zeigt den Hex-Dump des Programms.
Im Programm werden nur MS-DOS-Funktionen verwendet. Die Werte, die rechnerabhängig zu ändern sind, wurden am Programmanfang mit der EQU-Anweisung festgelegt und können geändert werden. Die Konstante int_vektor hat beim IBM-PC den Wert 4BH. Die Portadressen müssen der jeweils verwendeten Echtzeituhrenkarte angepaßt werden.

```
;---------- Vereinbarungen ----------
;
;Diese Werte sind maschinenabhängig:
;
port        equ 0f884h    ;Port fuer Uhrzeit (Siemens PC-D)
grafik      equ 8h        ;Grafik-Modus
int_vektor  equ 0f8h      ;Zeiger auf IRET (4bh beim IBM XT)
code segment
assume cs:code, ds:code   ;das Programm soll ein Com-File werden
org 100h

;---------- Residenter-Teil 1----------
start:          jmp eingabe       ;nach Aufruf
;----------:hier beginnt die Interruptabarbeitung ----------
zeit:           push ax           ;mindestbenötigte
                push ds           ;Register retten
                pushf
                push cs
                pop ds
                mov al, zaehler   ;eine Sekunde
                inc al            ;vergangen?
                cmp al, 20
                jz zeit_anzeige   ;ja, Zeit anzeigen
                mov zaehler, al   ;sonst neuen
                jmp ende          ;Wert speichern
zeit_anzeige:   mov zaehler, 0    ;Zaehler = 0
                mov al, anzeige   ;Anzeige ein?
                cmp al, 0
                jnz ende          ;nein, Ende
                push ds
                mov ax, 0040h
                mov ds, ax        ;BIOS-Datensegment
                mov al, [ds:0049h] ;
                pop ds
                cmp al, grafik    ;Grafik-Modus?
                jz ende           ;ja, keine Anzeige
                push bx           ;alle sonst benoetigte
                push cx           ;Register retten
                push dx
                push bp
                mov ah, 3
                xor bh, bh
                int 10h           ;Cursorposition lesen
                push dx           ;und retten
                mov dh, zeile
                mov dl, spalte
                mov ah, 2         ;Cursorposition fuer
                int 10h           ;Uhranz. schreiben
                mov dx, port      ;Stunden lesen
                xor ah, ah        ;Divident korrigieren
                in al, dx
                call display      ;Stunden anzeigen
                mov al, ':'       ;Doppelpunkt
                int 29h           ;ausgeben
                sub dx, 2
                xor ah, ah
                in al, dx         ;Minuten lesen
                call display      ;und anzeigen
                mov al, ':'       ;Doppelpunkt
                int 29h           ;ausgeben
                sub dx, 2
                xor ah, ah
                in al, dx         ;Sekunden lesen
                call display      ;und anzeigen
                pop dx            ;alte Cursorposition
                mov ah, 2         ;holen
                xor bh, bh        ;und schreiben
                int 10h
                pop bp            ;Register
                pop dx            ;zurückholen
                pop cx
                pop bx
ende:           popf              ;mindestbenötigte
                pop ds            ;Register
                pop ax            ;zurückholen
                iret              ;zurück zum INT 8
                                  ;
;---------- Unterprogramme Teil 1 ----------
;
;Anzeige ----------
;
display:    mov bl, 10            ;Wert 10er-Stelle
            div bl                ;errechnen (=AL)
            add al, 48
            mov bl, ah            ;Rest (=AH) retten
            int 29h               ;10er-Stelle ausgeben
```

Bild 1. Das Assemblerlisting ist durch die zahlreichen Kommentare gut dokumentiert

```
                mov al, bl
                add al, 48          ;Rest ausgeben
                int 29h
                ret
;---------------Datenteil Teil 1-----------------------------
zaehler db 0
zeile   db 0
spalte  db 72
anzeige db 0
;==============================================================
;--------------- nicht residenter Teil 2 ----------------------
;
eingabe:        push cs
                pop ds
                xor ax, ax
                mov es, ax              ;ES = Interrupt-Segment
                mov al, [es: 0070h]     ;ZEIT schon
                cmp al, int_vektor      ;installiert?
                jz installation         ; nein, installieren
                mov ax, [es: 0072h]     ;ES als Datensegment
                mov es, ax              ;der ZEIT-Routine
                call check_parameter
                int 20h                 ;Programm beenden
installation:   mov dx, offset titel    ;Titel auf Bildschirm
                mov ah, 9
                int 21h
                push ds                 ;ES fuer check_parameter
                pop es                  ;korrigieren
                call check_parameter
                mov anzeige, 0          ;Anzeige an
                mov dx, offset zeit     ;Int 1CH-Vektor auf
                mov ah, 25h             ;ZEIT-Routine
                mov al, 1ch             ;aendern
                int 21h
                mov ah, 31h
                mov dx, offset eingabe  ;ZEIT resident
                int 21h                 ;
;---------------- Unterprogramm Teil 2 ------------------------
;PARAMETER-CHECK -----------------------------------------------
check_parameter:    xor ch, ch          ;wegen LOOP
                    mov di, 80h
                    mov cl, [di]        ; CL = Parameteranzahl
                    cmp cl, 0
                    jnz parameter_0     ;
                    mov es: anzeige, 1  ;ja, Anzeige aus
                    ret
parameter_0:        inc di              ;naechstes Zeichen
                    mov al, [di]
                    cmp al, ' '         ;Leerzeichen?
                    jz next_parameter   ;ja, naechstes Zeichen
                    cmp al, ','         ;Komma?
                    jnz zahl_check      ;nein, Zahl interpr.
                    inc bh              ;ja, 2. Parameter (BH=1)
                    xor bl, bl
                    jmp next_parameter
zahl_check:         cmp al, 48          ;Eingabe = Ziffer
                    jl fehler_hinweis   ;< 0? ja, Fehler
                    cmp al, 57          ;> 9? ja, Fehler
                    jg fehler_hinweis
                    sub al, 48          ;ASCII = Zahl
                    cmp bh, 0           ;1. Parameter (BH = 0)?
                    jnz parameter_2     ;nein, 2. Parameter
parameter_1:        inc bl              ;Ziffernanz. inkrement.
                    cmp bl, 3           ;3 Ziffern sind zuviel
                    jz fehler_hinweis   ;Fehler
                    cmp bl, 1           ;1. Ziffer
                    jz para_1_1         ;ja, abspeichern
                    mov dl, al          ;nein, DL = 2. Ziffer
                    mov al, para1       ;1. Ziffer holen
                    mov dh, 10          ;mit 10 multiplizieren
                    mul dh
                    add al, dl          ;2. Ziffer addieren
para_1_1:           mov para1, al       ;1. Param. abspeichern
                    jmp next_parameter
parameter_2:        add bl, 1           ;gleicher Vorgang wie
                    cmp bl, 3           ;parameter_1
                    jz fehler_hinweis
                    cmp bl, 1
                    jz para_2_1
                    mov dl, al
                    mov al, para2
                    mov dh, 10
                    mul dh
                    add al, dl
para_2_1:           mov para2, al
next_parameter:     loop parameter_0    ;wenn DX <> 0
fehler_check:       cmp bh, 0           ;Komma vorhanden?
                    jz fehler_hinweis   ;nein, Fehler
                    mov ch, para1       ;1. Parameter holen
                    cmp ch, 23          ;> 23?
                    jg fehler_hinweis   ;ja, Fehler
                    mov cl, para2       ;2. Parameter holen
                    cmp cl, 72          ;> 72?
                    jg fehler_hinweis   ;ja, Fehler
                    jmp uebernahme      ;sonst Wert aendern
fehler_hinweis:     mov dx, offset fehler_text
                    mov ah, 9           ;fehler_text ausgeben
                    int 21h
                    jmp return
uebernahme:         mov al, para1
                    mov es: zeile, al
                    mov al, para2
                    mov es: spalte, al
                    mov es: anzeige, 0  ;Anzeige an
return:             ret
;
;---------------- Datenteil  Teil 2 ---------------------------
;
titel           db 'ECHTZEITUHR V1.10 10.10.86',13,'$'
fehler_text     db '?Falscher Parameter$'
para1           db 0
para2           db 0
code ends
end start
```

```
4ABF:0100  E9 8D 00 50 1E 9C 0E 1F-A0 8C 01 FE C0 3C 14 74   i..P.... ..@<.t
4ABF:0110  06 A2 8C 01 EB 61 90 C6-06 8C 01 00 90 A0 8F 01   ."..ka.F..... ..
4ABF:0120  3C 00 75 53 1E B8 40 00-8E D8 A0 49 00 1F 3C 08   <.uS.8@..X I..<.
4ABF:0130  74 45 53 51 52 55 B4 03-32 FF CD 10 52 8A 36 8D   tESQRU4.2.M.R.6.
4ABF:0140  01 8A 16 8E 01 B4 02 CD-10 BA 84 F8 32 E4 EC E8   .....4.M.:.x2dlh
4ABF:0150  29 00 B0 3A CD 29 83 EA-02 32 E4 EC E8 1C 00 B0   ).0:M).j.2dlh..0
4ABF:0160  3A CD 29 83 EA 02 32 E4-EC E8 0F 00 5A B4 02 32   :M).j.2dlh..Z4.2
4ABF:0170  FF CD 10 5D 5A 59 5B 9D-1F 58 CF B3 0A F6 F3 04   .M.]ZY[..XO3.vs.
4ABF:0180  30 8A DC CD 29 8A C3 04-30 CD 29 C3 00 00 48 00   0.\M).C.0M)C..H.
4ABF:0190  0E 1F 33 C0 8E C0 26 A0-70 00 3C F8 74 0B 26 A1   ..3@.@& p.<xt.&!
4ABF:01A0  72 00 8E C0 E8 23 00 CD-20 BA 71 02 B4 09 CD 21   r..@h#.M :q.4.M!
4ABF:01B0  1E 07 E8 15 00 C6 06 8F-01 00 BA 03 01 B4 25 B0   ..h..F....:..4%0
4ABF:01C0  1C CD 21 B4 31 BA 90 01-CD 21 32 ED BF 80 00 8A   .M!41:..M!2m?...
4ABF:01D0  0D 80 F9 00 75 07 26 C6-06 8F 01 01 C3 47 8A 05   ..y.u.&F....CG..
4ABF:01E0  3C 20 74 52 3C 2C 75 07-FE C7 32 DB EB 48 90 3C   < tR<,u.G2[kH.<
4ABF:01F0  30 7C 5F 3C 39 7F 5B 2C-30 80 FF 00 75 1D FE C3   0|_<9.[,0...u.C
4ABF:0200  80 FB 03 74 4D 80 FB 01-74 0B 8A D0 A0 A1 02 B6   .{.tM.{.t..P !.6
4ABF:0210  0A F6 E6 02 C2 A2 A1 02-EB 1C 90 80 C3 01 80 FB   .vf.B"!.k...C..{
4ABF:0220  03 74 2F 80 FB 01 74 0B-8A D0 A0 A2 02 B6 0A F6   .t/.{.t..P ".6.v
4ABF:0230  E6 02 C2 A2 A2 02 E2 A5-80 FF 00 74 15 8A 2E A1   f.B"".b%...t...!
4ABF:0240  02 80 FD 17 7F 0C 8A 0E-A2 02 80 F9 48 7F 03 EB   ..}.....".."yH..k
4ABF:0250  0B 90 BA 8D 02 B4 09 CD-21 EB 15 90 A0 A1 02 26   ..:..4.M!k..  !.&
4ABF:0260  A2 8D 01 A0 A2 02 26 A2-8E 01 26 C6 06 8F 01 00   ".. ".&"..&F...
4ABF:0270  C3 45 43 48 54 5A 45 49-54 55 48 52 20 56 31 2E   CECHTZEITUHR V1
4ABF:0280  31 30 20 31 30 2E 31 30-2E 38 36 0D 24 3F 46 61   10 10.10.86.$?Fa
4ABF:0290  6C 73 63 68 65 72 20 50-61 72 61 6D 65 74 65 72   lscher Parameter
4ABF:02A0  24 00 00 50
```

Bild 2. Mit Debug kann das Programm auch ohne Assembler eingeben werden

Berthold Weber

Das andere Chaos

In GfA-Basic wurde ein Programm für den Atari ST geschrieben, das beliebige Muster sehr schnell erzeugt.

Seit geraumer Zeit erfreuen sich sogenannte Fractal-Grafiken großer Beliebtheit bei Computerbesitzern. Unangenehm bei eigenen Experimenten mit den „Apfelmännchen" und „Seepferdchen" ist jedoch der große Rechenaufwand für die notwendige komplexe Fließkomma-Arithmetik. Anders das hier vorgestellte Verfahren: Es kommt mit CPU-freundlichen logischen Operationen aus, die vielfach auch in höheren Programmiersprachen verfügbar sind.

Grundbaustein des nachfolgend beschriebenen Grafikprogrammes ist eine Funktion, die ich „Outline" nennen möchte. Eine flächige Figur wird mit Outline in eine Umrißzeichnung umgewandelt. Die Programmierung ist einfach: Das Ausgangsbild wird um ein Pixel in x-Richtung versetzt logisch XOR auf sich selbst abgebildet. Das gleiche macht man in y-Richtung, ebenfalls um einen Punkt versetzt und addiert dann beide Abbildungen mit logisch OR.

Der eigentliche Trick, der ganz erstaunliche Grafiken liefert, ist nun der, die Outline-Funktion mehrfach auf das von ihr erzeugte Bild anzuwenden. Durch eine Art von Interferenzen entstehen schon aus einfachen Anfangsbildern mit nur zwei oder drei gesetzten Pixeln kaum vorhersehbare, teilweise symmetrische, teils chaotische Formen (*Bild 1*).

Eigenen Experimenten ist hier ein breiter Raum gegeben, da schon kleinste Abweichungen des Grundmusters total unterschiedliche Entwicklungen der Outline-Grafiken bewirken.

Andererseits lassen sich auch mit „höheren" Ausgangsobjekten wie Kreisen, geknickten Linien und Bögen interessante Mutationsfolgen erzielen. Besonders an Knickstellen oder geringen Unregelmäßigkeiten bilden sich schon nach wenigen Generationen kleine Wucherungen, die sich dann schnell ausbreiten und zum Teil wieder flächige Muster ergeben (*Bild 2*).

Das GfA-Basic-Programm (*Bild 3*) ist durch die Verwendung der Get- und Put-Befehle ausgesprochen schnell, so daß eine Assembler-Programmierung nicht notwendig ist. *Berthold Weber*

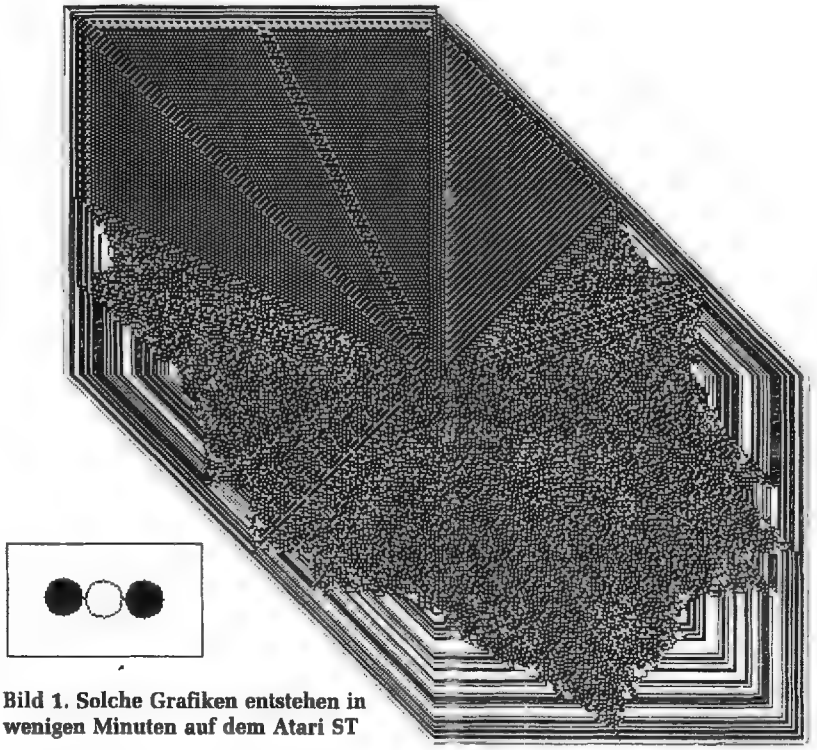


Bild 1. Solche Grafiken entstehen in wenigen Minuten auf dem Atari ST

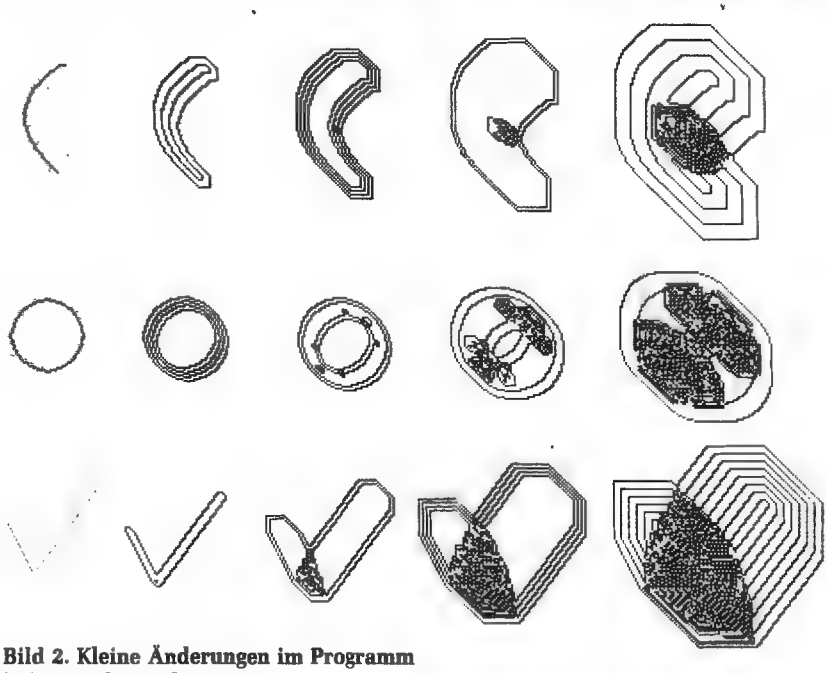


Bild 2. Kleine Änderungen im Programm haben große Wirkungen

```
Rem    (c) by Dipl.-Ing. Berthold Weber 1987
Rem          Schlörplatz 1
Rem          8480 Weiden i.d.OPf.
Rem
Rem   Outline-Grafiken für Atari ST (GfA-Basic) s/w
Rem
Rem ------- Hier die Ausgangszeichnung festlegen --------
Rem Circle 320,200,60
Rem Draw 300,200 To 350,210 To 370,230
Rem Draw 300,200 To 350,210 To 370,231
Box 280,100,390,280
Rem ------------------------------------------------------
Y=5
Dy=390
X=115
Dx=390
Gen=0
Box X-1,Y-1,X+Dx+1,Y+Dy+1
Box 520,5,630,95
Box 520,105,630,175
Text 530,20,"linke"
Text 530,40,"Maustaste"
Text 530,60,"für nächste"
Text 530,80,"Generation"
Text 530,120,"rechte"
Text 530,140,"Maustaste"
Text 530,160,"für Ende"
Do
  Exit If Mousek=2
  Get X,Y,X+Dx,Y+Dy,Hh1$
  Put X+1,Y,Hh1$,6
  Get X,Y,X+Dx,Y+Dy,Hh2$
  Put X,Y,Hh1$,3
  Put X,Y-1,Hh1$,6
  Put X,Y,Hh2$,7
  Inc Gen
  Text 5,20,"Gen.= "+Str$(Gen)
  While Mousek=0
  Wend
  Get X,Y,X+Dx,Y+Dy,Hh1$
  Put X-1,Y,Hh1$,6
  Get X,Y,X+Dx,Y+Dy,Hh2$
  Put X,Y,Hh1$,3
  Put X,Y+1,Hh1$,6
  Put X,Y,Hh2$,7
  Inc Gen
  Text 5,20,"Gen.= "+Str$(Gen)
  While Mousek=0
  Wend
Loop
```

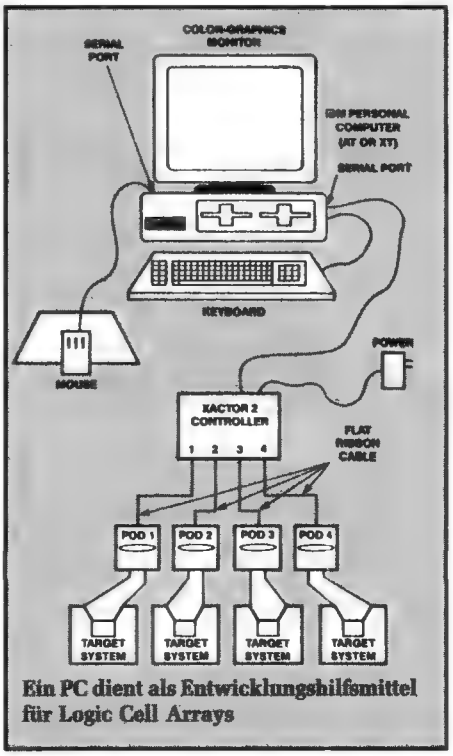
Bild 3. GfA-Basic-Programm zur Demonstration der Outline-Funktion

Programmierbare Logikbausteine

Die ersten programmierbaren Logikbausteine wurden vor einigen Jahren von MMI unter der Bezeichnung PAL (Programmable Array Logic) auf den Markt gebracht. Seit der Markteinführung der PALs wurden zahlreiche Verbesserungen an der Architektur dieser Bausteine vorgenommen, das Prinzip blieb jedoch das gleiche: Die Eingänge der PAL-Bausteine bestehen aus programmierbaren Und-Gliedern. Die Ausgänge dieser Und-Glieder werden mit Oder-Gliedern verbunden. Unter Anwendung der booleschen Algebra (Schaltalgebra) lassen sich mit dieser zweistufigen Logik (Und- und Oder-Glieder) sämtliche Funktionen der kombinatorischen Logik verwirklichen. Da man Und-Verknüpfungen als boolesches Produkt und Oder-Verknüpfungen als boolesche Summe bezeichnet, bilden die Ausgänge kombinatorischer PALs eine Summe von Produkten. Werden in PAL-Bausteinen Flipflops (PAL mit Register-Ausgang) integriert, die mit den (internen) Ausgängen der Und-/Oder-Arrays verbunden sind und Rückführungen von den Flipflop-Ausgängen zu dem Und-/Oder-Array vorgesehen, so lassen sich mit PALs auch Synchron-Schaltwerke wie z. B. Zähler aufbauen.

Ein Nachteil der PAL-Bausteine ist, daß bei komplexen Funktionen, sehr viele Flipflops und Rückführungen der Signale vom Ausgang zum Eingang gebraucht werden. Die Ein-/Ausgänge und die internen Flipflops sind schnell aufgebraucht, das programmierbare Und-Array ist jedoch nur zu einem Bruchteil ausgenutzt. Um auch für solche Fälle programmierbare Logikbausteine einsetzen zu können, hat MMI die Logic Cell Arrays (LCA) herausgebracht. Diese Bausteine bestehen aus einem Array von konfigurierbaren Logikblöcken, wobei jeder Logikblock, ähnlich einem PAL mit Registerausgang, aus einem programmierbaren Array zur Realisierung kombinatorischer Logik, einem Speicherelement und dem Rückführungszweig besteht. Die Verbindungen zwischen den konfigurierbaren Logikblökken sind in weiten Grenzen programmierbar. Zusammen mit den konfigurierbaren Ein-/Ausgangsblöcken können Schaltungen, die früher fast eine ganze Leiterplatte belegten oder als teurer kundenspezifischer Schaltkreis aufgebaut werden mußten, mit programmierbaren Logikbausteinen aufgebaut werden. Ein Schlüssel zum Einsatz dieser Technologie ist die Entwicklungssoftware, die auf einem PC oder einem AT läuft, der mit 640 KByte RAM, einem CGA-Adapter und einer Maus ausgerüstet sein muß. Viele herkömmliche Logikfunktionen, die früher z. B. mit Bausteinen aus der 74xx-Familie aufgebaut wurden, sind bereits als Makro in der Entwicklungssoftware enthalten. Die Makro-Bibliothek kann um eigene Makros erweitert werden. Mit einem grafischen Editor wird die gewünschte Schaltung eingegeben. Die Entwicklungsoftware überprüft, ob Entwurfsregeln verletzt wurden und berechnet Signalverzögerungen. Ein zusätzlich erhältlicher Simulator liefert das Zeit- und Logikverhalten der Schaltung in grafischer Form. Ein extern anschließbarer Controller, von MMI Xactor 2 genannt, ermöglicht die Emulation der Schaltung im Zielsystem.

Bei vielen Schaltungsänderungen wird man in Zukunft nicht mehr zum Lötkolben greifen müssen, sondern die Schaltungsänderung mit der Entwicklungs-Software durchführen.



Ein PC dient als Entwicklungshilfsmittel für Logic Cell Arrays

Nach Unterlagen von MMI

Gerhard Röhner

Apple-CP/M mit der Maus „booten“

Anlaß für diesen Artikel war die Forderung eines Darmstädter Apple-Händlers: Ich hatte bei ihm eine Apple-Maus und eine erweiterte 80-Zeichen-Karte erworben. Nach Einbau der beiden Karten in einen Apple-IIe blieb der Rechner beim Booten stets hängen. Nach Auskunft des Händlers ist dieses Problem durch Änderung von zwei Bytes im CP/M-Betriebssystem zu lösen. Allerdings wollte er 50 DM für ein Programm, das diese Patches durchführt.

Wie macht man es selbst? Beim Kaltstart legt CP/M ab F3B8H eine Kartentyp-Tabelle an. Dazu wird jeder Slot auf Vorhandensein einer Karte geprüft und der Kartentyp in der Tabelle vermerkt.

Löscht man während des Kaltstarts die Mauskarte aus der Tabelle, so bootet CP/M wieder normal.

Die Mauskarte wird anhand der Identifikationsbytes (Fn0CH)=($Cn0C)=$20 und (FnFBH)=($CnFB)=$D6 erkannt. Man sucht Slot n der Maus und löscht in der Kartentyp-Tabelle bei Adresse F3B8H+n den Eintrag für die Maus.

Das dazu nötige Programm MAUSSUCHEN (*Bild 1*) wird im Treiberbereich für LST: F200H...F280H untergebracht. Die Verbindung mit dem BIOS wird durch einen Patch der Karteninitialisierungs-Routine hergestellt. Sie beginnt in der 56-KByte-Version von CP/M 2.2 bei 0DAA2H mit LD DE,07. Der Ladebefehl wird durch ein JP F200H ersetzt.

Die Änderungen müssen auf die CP/M-System-Spuren und in den Speicher geschrieben werden. Diese Aufgabe erledigt das Programm MAUSBOOT (Bild 1). Die Änderungen kann man mit dem DDT-Set-Kommando direkt eingeben und mit SAVE 2 MAUSBOOT.COM auf Diskette abspeichern.

Eine gute Übersicht zur Speicherbelegung unter CP/M 2.2 (56-KByte-Version) gibt *Bild 2*.

```
; MAUSBOOT

0100: 01 00 02 CD 24 DA 01 00 00 CD 1E DA 01 02 00 CD
0110: 21 DA CD 2A DA 01 80 00 21 00 02 11 00 F2 ED B0
0120: 01 02 00 CD 1E DA 01 0D 00 CD 21 DA CD 27 DA 3E
0130: C3 32 22 02 32 A2 DA 3E 00 32 23 02 32 A3 DA 3E
0140: F2 32 24 02 32 A4 DA 0E 01 CD 2A DA C3 00 00 00

; MAUSSUCHEN

0200: 11 08 00 21 00 E8 25 1D 28 14 2E 0C 7E FE 20 20
0210: F5 2E FB 7E FE D6 20 EE 21 B8 F3 19 36 00 11 07
0220: 00 C3 A5 DA 00 00 00 00 00 00 00 00 00 00 00 00
```

Bild 1. Die Änderungen kann man mit dem DDT eingeben und abspeichern

```
      CP/M 2.2 Systemadressen    für Apple II   (56K-Vers.)

 Z80    6502

0000H  $1000  JMP WARMBT ; Sprung zur Warmstartinitialisierung
0003H  $1003  DS 1       ; IOBYTE
0004H  $1004  DS 1       ; Bezugslaufwerk und Benutzernummer
0005H  $1005  JMP BDOS   ; Sprung zum BDOS-Anfang
0008H  $1008  DS 40      ; Anlaufstellen für RST 1 bis RST 5
                         ; nicht benutzt
0030H  $1030  DS 8       ; RST 6, für CP/M reserviert, ungenutzt
0038H  $1038  JMP DDT    ; RST 7, wird von DDT/SID initialisiert
003BH  $103B  DS 5       ; Rest von RST 7, für CP/M reserviert
0040H  $1040  DS 16      ; frei für BIOS-Anwendungen
0050H  $1050  DS 12      ; für CP/M reserviert, ungenutzt
005CH  $105C  DS 36      ; Vorgegebener Dateibezeichner (FCB)
0080H  $1080  DS 128     ; Vorgegebener Datenbuffer (DMA)


C400H  $E400             ; *** CCP ***
C400H  $E400  JMP START  ; Normaler CCP-Aufruf
C403H  $E403  JMP CLEAR  ; CCP-Aufruf mit Pufferlöschung
C406H  $E406  DB 127     ; maximale Befehlspufferlänge
C407H  $E407  DS 1       ; Länge des vorliegenden Befehls
C408H  $E408  DS 8       ; acht Leerzeichen
C410H  $E410  DS 8       ; acht Leerzeichen
C418H  $E418             ; Copyright-Vermerk

CC00H  $EC00             ; *** BDOS ***
```

Bild 2. Übersicht über die Speicherbelegung von CP/M 2.2 in der 56-KByte-Version für den Apple-II

Literatur

[1] Microsoft Softcard, Volume I und II.
[2] AppleMaus-II-Benutzerhandbuch.

```
DA00H   $FA00               ; *** BIOS ***
DA00H   $FA00   RET         ; Cold Boot
DA03H   $FA03   JMP DACC    ; JMP WBOOT
DA06H   $FA06   JMP DB08    ; JMP CONST
DA09H   $FA09   JMP DB50    ; JMP CONIN
DA0CH   $FA0C   JMP DB43    ; JMP CONOUT
DA0FH   $FA0F   JMP DB66    ; JMP LIST
DA12H   $FA12   JMP DB75    ; JMP PUNCH
DA15H   $FA15   JMP DB87    ; JMP READER
DA18H   $FA18   JMP DD4B    ; JMP HOME
DA1BH   $FA1B   JMP DD6D    ; JMP SELDSK
DA1EH   $FA1E   JMP DD56    ; JMP SETTRK
DA21H   $FA21   JMP DD89    ; JMP SETSEC
DA24H   $FA24   JMP DD8E    ; JMP SETDMA
DA27H   $FA27   JMP DD93    ; JMP READ
DA2AH   $FA2A   JMP DDA3    ; JMP WRITE
DA2DH   $FA2D   XRA A! RET  ; LISTST nicht implementiert
DA30H   $FA30   MOV H,B     ; SECTRAN

DFFFH   $FFFF               ; Top of RAM

E000H   $C000   DS 4096     ; Apple I/O Seite
F000H   $0000   DS 512      ; 6502 Zeropage und Stack
F045H   $0045   DS 1        ; 6502 A-Register
F046H   $0046   DS 1        ; 6502 X-Register
F047H   $0047   DS 1        ; 6502 Y-Register
F048H   $0048   DS 1        ; 6502 P-Register
F049H   $0049   DS 1        ; 6502 S-Register
F200H   $0200   DS 512      ; I/O-Konfigurationsblock
F200H   $0200   DS 128      ; LST: Treiber          (Slot 1)
F280H   $0280   DS 128      ; PUN: und RDR: Treiber (Slot 2)
F300H   $0300   DS 128      ; TTY: Treiber          (Slot 3)
F380H   $0380   F34AH       ; Console Status Vector
F382H   $0382   F358H       ; Console Input  Vector 1 TTY: CRT:
F384H   $0384   DB29H       ; Console Input  Vector 2 UC1:
F386H   $0386   F35EH       ; Console Output Vector 1 TTY: CRT:
F388H   $0388   DC3EH       ; Console Output Vector 2 UC1:
F38AH   $038A   DD45H       ; Reader  Input  Vector 1 PTR:
F38CH   $038C   DD45H       ; Reader  Input  Vector 2 UR1: UR2:
F38EH   $038E   DD3FH       ; Punch   Output Vector 1 PTP:
F390H   $0390   DD3FH       ; Punch   Output Vector 2 UP1: UP2:
F392H   $0392   DD2BH       ; List    Output Vector 1 LPT:
F394H   $0394   DD2BH       ; List    Output Vector 2 UL1:
F396H   $0396   DS 1        ; Cursor Adress Offset   Software
F397H   $0397   DS 1        ; Lead in Character      Software
F398H   $0398   DS 1        ; Clear Screen           Software
F399H   $0399   DS 1        ; Clear to End of Page   Software
F39AH   $039A   DS 1        ; Clear to End of Line   Software
F39BH   $039B   DS 1        ; Set Normal Text Mode   Software
F39CH   $039C   DS 1        ; Set Invers Text Mode   Software
F39DH   $039D   DS 1        ; Home Cursor            Software
F39EH   $039E   DS 1        ; Address Cursor         Software
F39FH   $039F   DS 1        ; Move Cursor Up         Software
F3A0H   $03A0   DS 1        ; Cursor forward         Software

F3A1H   $03A1   DS 1        ; Cursor Adress Offset   Hardware
F3A2H   $03A2   DS 1        ; Lead in Character      Hardware
F3A3H   $03A3   DS 1        ; Clear Screen           Hardware
F3A4H   $03A4   DS 1        ; Clear to End of Page   Hardware
F3A5H   $03A5   DS 1        ; Clear to End of Line   Hardware
F3A6H   $03A6   DS 1        ; Set Normal Text Mode   Hardware
F3A7H   $03A7   DS 1        ; Set Invers Text Mode   Hardware
F3A8H   $03A8   DS 1        ; Home Cursor            Hardware
F3A9H   $03A9   DS 1        ; Address Cursor         Hardware
F3AAH   $03AA   DS 1        ; Move Cursor Up         Hardware
F3ABH   $03AB   DS 1        ; Cursor forward         Hardware

F3ACH   $03AC   DS 12       ; Keyboard Character Redefinition Table
F3B8H   $03B8   DS 1        ; Disk Count Byte
F3B9H   $03B9   DS 7        ; Card Type Table
F3C0H   $03C0               ; 6502 >\ Z80 Switch Routine
F3D0H   $03D0   DS 2        ; Adresse der aufzurufenden 6502 Routine
F3DEH   $03DE   DS 2        ; Adresse der SoftCard

F400H   $0400   DS 1024     ; Apple-Bildschirm
F800H   $0800   DS 1024     ; Diskettentreiber und Puffer
```

Neues vom Buchmarkt

EPROM-Programmer selbstgebaut

Wie jeder Computer zum Programmiergerät wird, beschreibt Hans-Joachim Blank in seinem jetzt erschienenen Buch. Die in allen Details beschriebene Programmierkarte für die Typen 2764, 27128 und 27256 kann an jeden Computer mit paralleler Schnittstelle angeschlossen werden. Die Maschinensprachroutinen liegen im 8085-Sourcecode vor.

Von Hans-Joachim Blank. Universeller EPROM-Programmer selbstgebaut, 61 Seiten, 14 Abbildungen, 16,80 DM, Franzis-Verlag, München, 1987.
ISBN 3-7723-2101-1

Atari ST: Arbeiten mit GEM

So komfortabel die grafische Bedieneroberfläche GEM des Atari ST auch ist, die Verwendung der zahlreichen Routinen in eigenen Programmen ist kompliziert. Mit zwei Büchern gibt der Sybex-Verlag jetzt beispielhaft die erforderliche Hilfestellung. Band 1 beschreibt die AES-Bibliothek, Band 2 die VDI-Bibliothek. Jedem Buch liegt eine Diskette mit vielen ablauffähigen Programmen und den entsprechenden C-Quellcodes bei.

Von Gerd Sender. Arbeiten mit GEM, Band 1: Die AES-Bibliothek, 320 Seiten, Diskette, 68 DM.
ISBN 3-88745-626-6
Von Holger Danielsson und Andreas Volkmann. Arbeiten mit GEM, Band 2: Die VDI-Bibliothek, 240 Seiten, Diskette, 68 DM, Sybex-Verlag, Düsseldorf, 1987.
ISBN 3-88745-627-0

C-Toolbox

Die typischen Komponenten einer Textverarbeitung stellt Edgar Huckert in seinem Buch in Form von C-Modulen transparent dar. Besonders interessant sind Silbentrenn- und Spellcheck-Routinen. Ein komplettes Textverarbeitungsprogramm liegt dem Buch bei.

Von Edgar Huckert. C-Toolbox Textverarbeitung, 273 Seiten, zwei Disketten, 69 DM, Verlag Markt & Technik AG, Haar, 1986.
ISBN 3-89090-364-9

Stephan Laage

Grafik-Paket in C

Darstellung von Funktionen und Meßwerten

Mit einem universellen Grafikprogramm ist es kein Problem, die zahlreichen Werte einer Meßreihe ansprechend darzustellen. Als Ausgabegerät kann ein Plotter, ein Matrixdrucker oder ein Monitor dienen. Die Berechnung von Spline-Funktionen ist enthalten.

Nicht nur, wer sich mit wissenschaftlichen Fragestellungen beschäftigt, steht oft vor dem Problem, Meßwerte oder mathematische Funktionen in einem zweidimensionalen Koordinatensystem darzustellen. Während man früher zu Millimeterpapier und Lineal griff, wird heute meist der inzwischen überall vorhandene Mikrocomputer eingesetzt. Doch allzu oft wird dann „auf die Schnelle" ein kleines Programm entworfen, daß das Gewünschte auf den Plotter bringen soll. Dabei zeigt jedoch die Erfahrung, daß aus dem kleinen Programm eine unangenehm lange Sitzung werden kann.
Wie schön, wenn man in solchen Fällen auf fertige Routinen zurückgreifen kann, die die Achsenskalierung, das Zeichnen der Achsen und Funktionswerte übernehmen; so, wie es auf Großrechner schon lange üblich ist.
Hier soll ein solches Paket vorgestellt werden, das es ermöglicht, mit einem wenige Zeilen umfassenden Programm sehr ansprechende Grafiken zu erzeugen. Dabei wird eine automatische Achsenskalierung und die Berechnung einer Spline-Funktion angeboten, die diskrete Werte durch eine glatte Funktion miteinander verbindet.

Die Umgebung

Dieses Paket wurde in der Programmiersprache C unter dem Betriebssystem OS-9 entwickelt. C bietet gegenüber (Turbo-)Pascal den Vorteil, daß ein komplettes Programm aus mehreren Dateien zusammengesetzt werden kann, wovon hier ausgiebig Gebrauch gemacht wird. Da, abgesehen von vier elementaren mathematischen Routinen, keine Funktionen verwendet werden, die über den von Kernighan und Ritchie (1) definierten Standard hinausgehen, sollte (fast) jeder andere C-Compiler auch unter anderen Betriebssystemen die Programme verdauen. Bei den vier mathematischen Routinen handelt es sich um
– double log10(x),
– double floor(x), liefert den größten, ganzzahligen Wert, der kleiner als x ist,
– double pow(x,y), liefert x hoch y.
– double fabs (x), liefert abs (x)
Um den Umfang der Listings möglichst klein zu halten und betriebssystemspezifische Dinge herauszuhalten, wurde auf die Behandlung von Fehlern vollständig verzichtet. Wer intensiver mit diesem Paket arbeiten möchte, sollte sich zu diesem Punkt eigene Gedanken machen – denn beim Arbeiten mit Zeigern ist schnell etwas „lebenswichtiges" im Rechner „abgeschossen".

Die Datei „scale.c"

Bild 1 umfaßt fünf Funktionen, die das komfortable Zeichnen von Daten in zweidimensionalen Koordinatensystemen ermöglichen. Die einzelnen Funktionen sollen kurz vorgestellt werden:
SCALE: Diese Funktion führt eine automatische Skalierung für ein eindimensionales Datenfeld durch. Die X- und die Y-Werte der zu zeichnenden Daten müssen getrennt mit SCALE() behandelt werden. SCALE() übernimmt drei Parameter: einen Zeiger auf das Datenfeld (DATA), die Anzahl der Werte im Feld (N) und die Zahl der gewünschten Achsenmarkierungen (NM). Die Funktion berechnet daraus die drei Skalierungskoeffizienten: den Wert des ersten Markierungsstriches für die entsprechende Achse (START), die Differenz zwischen den Werten zweier benachbarter Markierungsstriche (STEP) und die Anzahl der

```
/* SCALE ***************************************************************/
/* Paket zum Zeichnen von Funktionen und Daten in zweidimensionalen    */
/* Koordinatenkreuzen, enthaelt die Funktionen                         */
/*                                                                     */
/*      SCALE       automatische Skalierung eines Datenfeldes          */
/*      AXIS        Zeichnen von Achsen mit Markierungen               */
/*      RASTER      Zeichnen von Rastern                               */
/*      LINE        Zeichnen von Datenfeldern durch Linien oder Symbole*/
/*      SPLINE      natuerliche, kubische Splineinterpolation          */
/*                                                                     */
/* Stephan Laage                                        12. Oktober '86*/
/*                                    letzte Aenderung  29. Dezember '86*/
/***********************************************************************/


#include <math.h>
#define RND 1.0E-10
#define MAXS 50

/* Externe Deklarationen ***********************************************/
extern init(), move(), draw(), rotate(), print(), color(), symbol();
extern double chwe, chhe;


/* SCALE ***************************************************************/
/* berechnet die Skalierungskoeffizienten fuer ein eindimensionales DOUBLE */
/* Feld, die zum Zeichnen einer Koordinatenachse benoetigt werden: Wert */
/* der ersten Achsenmarkierung, Differenz zwischen zwei Markierungen und */
/* Anzahl der Markierungen                                             */

scale(data,n,nm)
    double *data;       /* zeigt auf das Datenfeld                     */
    int n;              /* Anzahl der Werte im Datenfeld               */
    int nm;             /* gewuenschte Zahl der Achsenmarkierungen     */
{
    double min,max,start,step,fak;
    int i,intstep;
```

Bild 1. Die Datei scale.c enthält alle Funktionen, die zum komfortablen Zeichnen von mathematischen Funktionen oder Meßwerten mit skalierten Koordinatenachsen notwendig sind

```
    min = *data;
    max = *data;
    for (i = 1; i < n; ++i) {
        ++data;
        if (*data < min) min = *data;
        if (*data > max) max = *data;
    };
    if (min > 0.0 && min - (max - min) / 4 < 0.0) min = 0.0;
    step = (max-min)/nm;
    fak = pow(10.0,floor(log10(step)));
    intstep = (int) (step/fak);
    switch (intstep) {
        case 1: intstep = 2;
                break;
        case 2:
        case 3:
        case 4: intstep = 5;
                break;
        case 5:
        case 6:
        case 7: intstep = 8;
                break;
        case 8:
        case 9: intstep = 10;
    };
    step = intstep * fak;
    if (min <= 0.0 && max > 0.0) {
        start = 0.0;
        while (start > min) start -= step;
        } else {
            fak *= 10.0;
            start = floor(min/fak)*fak;
            while (start + step <= min) start += step;
        };
        nm = 1;
        while (start + nm * step < max) ++nm;
        ++data;
        *(data++) = start;
        *(data++) = step;
        *(data) = (double) nm;
    };
    /* AXIS ************************************************************************/
    /* zeichnet eine Koordinaten-Achse mit Markierungen                            */

    axis(xp,yp,len,start,step,nmark,atyp,mtyp,titel)
        double xp,yp;       /* Startpunkt (mm)                                     */
        double len;         /* Achsenlaenge (mm)                                   */
        double start,step;  /* Achsenmarkierung: 1. Wert und Schrittweite          */
        double nmark;       /* Anzahl der Achsenmarkierungen                       */
        int atyp;           /* Achsentyp   0: X-Achse, 1: Y-Achse                  */
        int mtyp;           /* Markierungstyp   0: unten/rechts, 1: oben/links     */
        char *titel;        /* Achsenbeschriftung (wird zentriert)                 */
    {
        int i, nks, expo = 0;
        double fak = len/nmark, dis1, dis2, dis3, dis4;
        char str[20], form[5];

        if (fabs(start) + nmark * step < 0.001)
            while (fabs(start) + nmark * step < 1.0) {
                start *= 10.0;
                step *= 10.0;
                --expo;
            }
        else if (fabs(start) + nmark * step > 10000.0)
            while (fabs(start) + nmark * step > 10.0) {
                start /= 10.0;
                step /= 10.0;
                ++expo;
            };
        strcpy(form,"%.xf");
        nks = (int) -floor(log10(step+RND));
        if (nks < 0) nks = 0;
        form[2] = (char) nks+'0';

        if (mtyp == 0) {
            dis1 = -0.5*chhe;
            dis2 = -2.0*chhe;
            dis3 = -3.0*chhe;
            dis4 = -3.5*chhe;
        } else {
            dis1 = 0.5*chhe;
            dis2 = 1.0*chhe;
            dis3 = 3.0*chhe;
            dis4 = 2.5*chhe;
        };
        if (atyp == 0) {
            rotate(0);
            move(xp+len/2-strlen(titel)*chwe/2,yp+dis4);
        } else {
            rotate(1);
            dis1 = -dis1;
            dis2 = -dis2;
            dis3 = -dis3;
            dis4 = -dis4;
            move(xp+dis4,yp+len/2-strlen(titel)*chwe/2);
        };
        print(titel);

        start = start + RND;
        for (i = 0; i < nmark+0.1; ++i) {
            sprintf(str,form,start+i*step);
            if (atyp == 0) {
                move(xp+i*fak,yp);
                draw(xp+i*fak,yp+dis1);
                move(xp+i*fak-chwe*strlen(str)/2,yp+dis2);
                print(str);
                if (i < nmark-0.9) {
                    move(xp+i*fak,yp);
                    draw(xp+(i+1)*fak,yp);
                }
            } else {
                move(xp,yp+i*fak);
                draw(xp+dis1,yp+i*fak);
                move(xp+dis2,yp+i*fak-chwe*strlen(str)/2);
                print(str);
                if (i < nmark-0.9) {
                    move(xp,yp+i*fak);
                    draw(xp,yp+(i+1)*fak);
                }
            };
        };
        if (expo != 0) {
            sprintf(str,"%d",expo);
            if (atyp == 0) {
                move(xp+len-7*chwe,yp+dis4);
                print("*10");
                move(xp+len-4*chwe,yp+dis3);
            } else {
                move(xp+dis4,yp+len-7*chwe);
                print("*10");
                move(xp+dis3,yp+len-4*chwe);
            };
            print(str);
        };
    }
```

```
/* RASTER ************************************************************/
/* zeichnet ein Raster in das durch SCALE definierte Koordinatenkreuz */

raster(xp,yp,xl,yl,xn,yn)
    double xp,yp;       /* untere, linke Eckposition des Kreuzes (mm)  */
    double xl,yl;       /* Laenge der X- und der Y-Achse (mm)          */
    double xn,yn;       /* Anzahl der Markierungen                     */
{
    double xfak = xl / xn,
           yfak = yl / yn;
    int i, sw = 1;

    color(2);
    for (i = 0; i < xn + 0.1; ++i)
        if (sw) {
            move(xp+i*xfak,yp);
            draw(xp+i*xfak,yp+yl);
            sw = 0;
        } else {
            move(xp+i*xfak,yp+yl);
            draw(xp+i*xfak,yp);
            sw = 1;
        };
    for (i = 0; i < yn + 0.1; ++i)
        if (sw) {
            move(xp,yp+i*yfak);
            draw(xp+xl,yp+i*yfak);
            sw = 0;
        } else {
            move(xp+xl,yp+i*yfak);
            draw(xp,yp+i*yfak);
            sw = 1;
        };
    color(0);
}


/* LINE **************************************************************/
/* zeichnet einen Kurvenzug in das durch SCALE definierte Koordinatenkreuz */

line(xdata,ydata,n,xp,yp,xl,yl,symb)
    double *xdata,*ydata;   /* Datenfelder mit X- und Y-Werten         */
    int n;                  /* Anzahl der Werte in xdata/ydata         */
    double xp,yp;           /* unterer, linker Eckpunkt des Kreuzes (mm) */
    double xl,yl;           /* Laenge der Achsen (mm)                  */
    int symb;               /* < 0: nur Datenpunkte, > 0: mit Linienzug */
{
    double xst = xdata[n], yst = ydata[n];
    double xf,yf;
    int i;

    xf = xl/xdata[n+1]/xdata[n+2];
    yf = yl/ydata[n+1]/ydata[n+2];
    if (symb >= 0) {
        move(xp + (*xdata - xst) * xf, yp + (*ydata - yst) * yf);
        for (i = 0; i < n-1; ++i) {
            if (symb > 0) symbol(symb);
            ++xdata;
            ++ydata;
            draw(xp + (*xdata - xst) * xf, yp + (*ydata - yst) * yf);
        };
        if (symb > 0) symbol(symb);
    } else
        for (i = 0; i < n; ++i) {
            move(xp + (*xdata - xst) * xf, yp + (*ydata - yst) * yf);
            symbol(-symb);
            ++xdata;
            ++ydata;
        };
}


/* SPLINE ************************************************************/
/* berechnet aus diskreten Stuetzstellen eine kontinuierliche Spline-Funk- */
/* tion. Die Anzahl der Stuetzpunkte ist begrenzt.                    */

spline(xs,ys,ns,xf,yf,nf)
    double *xs,*ys; /* Felder mit den Koordinaten der Stuetzpunkte     */
    int ns;         /* Anzahl der Werte in xs und ys                   */
    double *xf,*yf; /* Felder, in denen die Funktionswerte abgelegt werden */
    int nf;         /* Anzahl der gewuenschten Spline-Funktionswerte   */
{
    static double a[MAXS],b[MAXS],c[MAXS],d[MAXS],
                  l[MAXS],m[MAXS],p[MAXS],h[MAXS],y2[MAXS];
    double xx,xstep;
    int i,k;

    if (ns > MAXS) ns = MAXS;                   /* maximal MAXS Stuetzpunte */
    for (i = 0; i <= ns - 2; ++i)
        h[i] = xs[i+1] - xs[i];
    for (i = 1; i <= ns - 2; ++i) {
        a[i] = 2 * (h[i-1] + h[i]);
        b[i] = h[i];
        c[i] = h[i];
        d[i] = 6 * ((ys[i] - ys[i-1]) / h[i-1] - (ys[i+1] - ys[i]) / h[i]);
    };
    m[1] = a[1];
    for (i = 1; i <= ns - 3; ++i) {
        l[i] = c[i] / m[i];
        m[i+1] = a[i+1] - l[i] * b[i];
    };
    p[1] = d[1];
    for (i = 2; i <= ns - 2; ++i)
        p[i] = d[i] - l[i-1] * p[i-1];
    y2[ns-2] = -p[ns-2] / m[ns-2];
    for (i = ns - 3; i >= 1; --i)
        y2[i] = -(p[i] + b[i] * y2[i+1]) / m[i];
    y2[0] = 0.0;
    y2[ns-1] = 0.0;
    for (i = 0; i <= ns - 2; ++i) {
        a[i] = (y2[i+1] - y2[i]) / (6 * h[i]);
        b[i] = y2[i] / 2;
        c[i] = (ys[i+1] - ys[i]) / h[i] - h[i] * (y2[i+1] + 2 * y2[i]) / 6;
        d[i] = ys[i];
    };
    xf[0] = xs[0];
    yf[0] = ys[0];
    xstep = (xs[ns-1] - xs[0]) / (nf - 1);
    k = 0;
    for (i = 1; i <= nf - 2; ++i) {
        xf[i] = xs[0] + i * xstep;
        if (xf[i] > xs[k+1]) ++k;
        xx = xf[i] - xs[k];
        yf[i] = a[k]*xx*xx*xx + b[k]*xx*xx + c[k]*xx + d[k];
    };
    xf[nf-1] = xs[ns-1];
    yf[nf-1] = ys[ns-1];
}
```

```
/* GPLOT ***************************************************************/
/* Graphik-Ankopplung C.ITOH-Plotter                                   */
/* Stephan Laage                                        Dezember '86   */
/***********************************************************************/

#include <stdio.h>

double chhe =         2.40,     /* Zeichen-Hoehe in Millimeter          */
       chwe =         1.60;     /* Zeichen-Breite                       */

double xfak = 2950.0/295.0,     /* 2950 Schritte auf 295.0 Millimeter   */
       yfak = 2100.0/210.0;     /* 2100 Schritte auf 210.0 Millimeter   */

FILE *fopen(), *pr;             /* "pr" = Druckerport                   */

init()
{
    pr = fopen("/p","w");       /* Druckerport oeffnen                  */
    fprintf(pr,"G1\n");         /* Schrittweite 0.1 mm                  */
    fprintf(pr,"H\n");          /* Stift in "Home"-Position             */
}

draw(x,y)
double x,y;
{
    fprintf(pr,"D %.0f,%.0f\n", x * xfak, y * yfak);
}                                       /* fahren mit gesenktem Stift */

move(x,y)
double x,y;
{
    fprintf(pr,"M %.0f,%.0f\n", x * xfak, y * yfak);
}                                       /* fahren mit gehobenem Stift */

print(str)
char *str;
{
    fprintf(pr,"P%s\n", str);                       /* Text ausgeben */
}

rotate(dir)
int dir;
{
  switch (dir) {
    case 0: fprintf(pr,"Q0\n"); break;      /* waagerecht, nach rechts  */
    case 1: fprintf(pr,"Q3\n"); break;      /* senkrecht, aufwaerts     */
    case 2: fprintf(pr,"Q2\n"); break;      /* waagerecht, nach links   */
    case 3: fprintf(pr,"Q1\n");             /* senkrecht, abwaerts      */
  };
}

color(c)
int c;
{
    fprintf(pr,"C%d\n", c);                 /* waehlt die Farbe         */
}

symbol(i)
int i;
{
    switch (i) {
        case 1: fprintf(pr,"N5\n"); break;      /* Symbol 5 = Kreis     */
        case 2: fprintf(pr,"N2\n"); break;      /* Symbol 2 = Quadrat   */
        case 3: fprintf(pr,"N1\n");             /* Symbol 1 = Karo      */
    }
}
```

Bild 2. Ein Beispiel für die Anbindung eines Plotters an die Datei scale.c. Hier handelt es sich um einen Plotter vom Typ C.Itoh, der eine Schrittweite von 0,1 mm aufweist

```
/* GSCREEN *************************************************************/
/* Graphik-Ankopplung fuer NEC7220-Graphikeinheit                      */
/* Aufloesung:  X-Richtung: 768 Punkte      (Spreizung auf DIN A 4)    */
/*              Y-Richtung: 576 Punkte                                 */
/*                                                                     */
/* Stephan Laage                                        Dezember '86   */
/***********************************************************************/

/* Externe Deklaration der Graphik-Systemroutinen **********************/
extern g_init(), g_move(), g_draw(), g_dir(), g_lpat(), g_print();

double chhe =         4.80,     /* Schriftzeichen-Hoehe in Millimeter   */
       chwe =         3.10;     /* Schriftzeichen-Breite                */

double xfak =  768.0/295.0,     /* 768 Pixel auf 295 Millimeter         */
       yfak = -576.0/210.0,     /* 576 Pixel auf 210 Millimeter         */
       yoff =       576.0;     /* Ursprung fuer Y                      */
int    adir =           2,     /* aktuelle Zeichenrichtung             */
       xpi  =           0,     /* aktuelle X-Position (Pixel)          */
       ypi  =           0;     /* aktuelle Y-Position (Pixel)          */

init()
{
    g_init(1);          /* initialisiert die Graphik-Ausgabeeinheit     */
}

move(x,y)
double x,y;
{
    xpi = (int) (x * xfak + 0.5);
    ypi = (int) (y * yfak + yoff + 0.5);
    g_move(xpi,ypi);    /* setzt den Graphik-Cursor                     */
}

draw(x,y)
double x,y;
{
    xpi = (int) (x * xfak + 0.5);
    ypi = (int) (y * yfak + yoff + 0.5);
    g_draw(xpi,ypi);    /* zeichnet eine Gerade von der Cursor-Position */
}

print(str)
char *str;
{
    g_print(str);       /* gibt Text aus                                */
    g_lpat(-1);         /* setzt Linienmuster zurueck (= $ffff)         */
}

rotate(dir)
int dir;
{
    switch (dir) {
        case 0: adir = 2; break;    /* waagerecht, nach rechts          */
        case 1: adir = 4; break;    /* senkrecht, aufwaerts             */
        case 2: adir = 6; break;    /* waagerecht, nach links           */
        case 3: adir = 0;           /* senkrecht, abwaerts              */
    };
    g_dir(adir);        /* setzt Schreib- und Zeichenrichtung           */
}

color(c)
int c;
{
```

Bild 3. Mit dieser Datei wird gezeigt, wie die Ausgaben von scale.c auf einem Grafik-Monitor ausgegeben werden. Dabei werden elementare Zeichenroutinen und Routinen zur Ausgabe von Schriftzeichen eingesetzt

```
    switch (c) {
        case 0: g_lpat(-1); break;          /* durchgezogene Linie       */
        case 1: g_lpat(0xaaaa); break;      /* gestrichelte Linie        */
        case 2: g_lpat(0x8888); break;      /* gepunktelte Linie         */
        case 3: g_lpat(0x8080);             /* stark gepunktelte Linie   */
    };
}

symbol(i)
int i;
{
    switch (i) {
        case 1: g_circle(6);        /* zeichnet einen Kreis          */
                break;
        case 2: g_move(xpi-5,ypi+5);
                g_dir(2);
                g_box(10,10);       /* zeichnet ein Rechteck         */
                g_move(xpi,ypi);
                g_dir(adir);
                break;
        case 3: g_move(xpi-7,ypi);
                g_dir(1);
                g_box(7,7);         /* zeichnet ein Karo             */
                g_move(xpi,ypi);
                g_dir(adir);
                break;
    };
}
```

```
/* BEISP1 ************************************************************/
/* fuer das Paket SCALE.C                              Stephan Laage */
/*********************************************************************/


#define MAXS 11
#define MAXF 100

/* Externe Deklarationen                                             */

extern scale(), axis(), raster(), line(), spline();
extern init(), move(), draw(), rotate(), print();


main()
{
    static double xf[MAXF+3], yf[MAXF+3];
    static double xs[MAXS+3] =
        {2.5, 3.3, 4.0, 4.5, 5.1, 5.4, 6.0, 6.45, 7.0, 7.8, 8.6};
    static double ys[MAXS+3] =
        {1.02E-5, 1.55E-5, 2.73E-5, 4.2E-5, 5.76E-5, 7.2E-5, 6.38E-5,
         2.8E-5, 8.9E-6, 1.4E-6, 5.0E-7};
    double xp = 30.0, yp = 35.0, xl = 240.0, yl = 150.0;
    int i;

    init();
    scale(xs,MAXS,12);
    scale(ys,MAXS,8);
    axis(xp,yp,xl,xs[MAXS],xs[MAXS+1],xs[MAXS+2],0,0,"pH");
    axis(xp,yp,yl,ys[MAXS],ys[MAXS+1],ys[MAXS+2],1,1,
                          "Reaktionsgeschwindigkeit [mol/min]");
    raster(xp,yp,xl,yl,xs[MAXS+2],ys[MAXS+2]);
    line(xs,ys,MAXS,xp,yp,xl,yl,-2);
    spline(xs,ys,MAXS,xf,yf,MAXF);
    for (i = 0; i < 3; ++i) {
        xf[MAXF+i] = xs[MAXS+i];
        yf[MAXF+i] = ys[MAXS+i];
    };
    line(xf,yf,MAXF,xp,yp,xl,yl,0);
    move(xp-20.0,yp-25.0);
    rotate(0);
    print("pH-Abhaengigkeit der sauren Phosphatase");
    move(xp-25.0,yp-30.0);
    draw(xp-25.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp-30.0);
    draw(xp-25.0,yp-30.0);
}
```

Bild 4. Beispiel für den Einsatz der Datei scale.c. 10 Meßwerte wurden eingetragen und durch die Spline-Funktion verbunden

Markierungsstriche, die die Achse tragen wird. Der letzte Wert kann von der gewünschten Markierungszahl etwas abweichen, je nach dem, wie es sich als günstig erweist. Diese drei Werte werden in dem Datenfeld selbst abgelegt, und zwar in den Zellen, die auf die eigentlichen Daten folgen. Entsprechend muß das Feld DATA zusätzlichen Platz für mindestens drei weitere Werte haben. Dafür hat der Aufrufer Sorge zu tragen, andernfalls ist der Crash vorprogrammiert.
Die Autoskalierung ermittelt für STEP Werte vom Typ 1, 2, 5, 8 oder 10, jeweils multipliziert mit einer geeigneten Zehnerpotenz. Anschließend wird START auf einen möglichst „glatten" Wert gesetzt, der kleiner als der kleinste im Datenfeld vorhandene Wert ist. Schließlich wird NM so gewählt, daß die Achse mindestens bis zum größten Datenwert reicht. In der Regel liefert SCALE() Achsen, die „vernünftig" skaliert sind.

AXIS: Diese Funktion zeichnet die Achse, wobei erst jetzt die Position (XP und YP) und die Achsenlänge (LEN) festgelegt werden. AXIS() übernimmt außerdem die drei durch SCALE() ermittelten Werte. Schließlich kann man noch wählen, ob es eine senkrechte oder eine waagerechte Achse werden soll (ATYP) und ob die Achsenbeschriftung unter bzw. rechts von der Achse oder über bzw. links von der Achse angebracht werden soll (MTYP). Dann ist noch ein Text anzugeben (TITEL), der zentriert an die Achse geschrieben wird.
Die Funktion schaltet bei Markierungswerten, die größer als 10 000 oder kleiner als 0,001 sind, auf eine Beschriftung im wissenschaftlichen Format mit einem „10 hoch X" am Ende der Achse um. Die funktionsinternen Variablen DIS1 bis DIS4 dienen der richtigen Positionierung der Beschriftungen. Im Feld FORM wird ein Formatstring abgelegt, der die geeignete Zahl von Nachkommastellen für die Achsenmarkierungen wählt und beim Aufruf der Funktion SPRINTF() gebraucht wird.

RASTER: Diese Funktion zeichnet ein Raster, nach Möglichkeit in einer anderen Farbe. Um bei Verwendung eines Plotters die abzufahrende Wegstrecke klein zu halten, werden die aufeinanderfolgenden Rasterstriche in wechselnden Richtungen abgefahren.

LINE: Mit dieser Funktion werden die Datenwerte eingezeichnet. LINE() erwartet zwei eindimensionale DOUBLE-Datenfelder, in denen die X- und Y-Koordi-

```
#include <math.h>
#define MAXP 200

/* Externe Deklarationen                                          */

extern init(), scale(), axis(), raster(), spline();
extern move(), draw(), rotate(), print();
main()
{
    static double x[MAXP+3], y[MAXP+3];
    double x1 = -5.0, x2 = 10.0, xstep;
    double xp = 20.0, yp = 20.0, xl = 240.0, yl = 150.0;
    int i;

    xstep = (x2 - x1) / (MAXP - 1);
    for (i = 0; i < MAXP; ++i) {
        x[i] = x1 + i * xstep;
        y[i] = 2 * sin(x[i]) + sin(2*x[i] + PI/2);
    };
    init();
    scale(x,MAXP,15);
    scale(y,MAXP,10);
    axis(xp,yp-y[MAXP]*yl/y[MAXP+1]/y[MAXP+2],xl,
                              x[MAXP],x[MAXP+1],x[MAXP+2],0,0,"X-Achse");
    axis(xp-x[MAXP]*xl/x[MAXP+1]/x[MAXP+2],yp,yl,
                             y[MAXP],y[MAXP+1],y[MAXP+2],1,1,"Y-Achse");
    raster(xp,yp,xl,yl,x[MAXP+2],y[MAXP+2]);
    line(x,y,MAXP,xp,yp,xl,yl,0);
    move(xp,yp+yl+10);
    rotate(0);
    print("f(x) = y = 2 sin x + sin (2 x + pi/2)");
    move(xp-10.0,yp-10.0);
    draw(xp-10.0,yp+yl+25.0);
    draw(xp+xl+10.0,yp+yl+25.0);
    draw(xp+xl+10.0,yp-10.0);
    draw(xp-10.0,yp-10.0);
```

Bild 5. Mit scale.c können beliebige mathematische Funktionen gezeichnet werden. Die Achsenlage ist frei wählbar

naten abgelegt sind. In beiden Feldern müssen auf die eigentlichen Daten die drei Skalierungskoeffizienten folgen, wie sie durch SCALE dort abgelegt werden. Natürlich können diese Werte auch „von Hand" dort eingetragen werden.

Mit dem Wert SYMB kann man wählen, ob ein Linienzug gezeichnet werden soll (SYMB = 0), ob die Datenpunkte durch Symbole markiert werden sollen (SYMB < 0) oder ob beides gezeichnet werden soll (SYMB > 0). Ist SYMB ungleich Null, dann wird durch den absoluten Wert von SYMB ein Symbol zur Markierung der Datenpunkte ausgesucht, etwa 1 = Kreis, 2 = Karo und 3 = Quadrat.

SPLINE: Diese Funktion zeichnet nichts. Stattdessen erwartet sie zwei Felder mit diskreten Stützpunkten (XS, YS), z. B. Meßwerten, und berechnet daraus eine natürliche, kubische Spline-Funktion, deren Funktionswerte in den Feldern XF und YF abgelegt werden. Mit NS wird die Anzahl der Stützpunkte und mit NF die gewünschte Anzahl der Spline-Funktionswerte übergeben. Dabei muß NF unbedingt größer als NS sein.

Der hier verwendete Algorithmus ist in der numerischen Mathematik verbreitet und unter anderem bei (2) nachzulesen. Um den Algorithmus einigermaßen überschaubar zu halten, wurde auf die Verwendung von Pointern vollständig verzichtet, was vielleicht ein bißchen Rechenzeit gespart hätte. Stattdessen wird nur die indizierte Feld-Adressierung eingesetzt.

Die Peripherie

Nun bleibt die Frage, auf welche Art und Weise die erstellte Grafik dem Betrachter zugänglich gemacht werden soll.

Prinzipiell sind als Ausgabegeräte (Grafik-)Monitor, Matrixdrucker oder Plotter denkbar. Um das Paket „scale.c" universell verwenden zu können, muß es mit einer Datei zusammengebunden werden, die die Ankopplung an das spezifische Ausgabegerät übernimmt. Zuerst einmal muß diese Datei zwei DOUBLE-Werte (CHHE und CHWE) enthalten, in denen die Breite und die Höhe der Schriftzeichen in Millimetern enthalten ist. Diese Werte werden von der Funk-

```
#include <math.h>

#define MAXN 150
#define LN2 0.6931471806

/* Externe Deklarationen                                          */

extern scale(), axis(), raster(), line();
extern init(), move(), draw(), print(), rotate();
main()
{
    static double time[MAXN+3],
                  ca45[MAXN+3], s35[MAXN+3], i125[MAXN+3], p32[MAXN+3];
    double lca45 = LN2/165.0, ls35 = LN2/87.2,
           li125 = LN2/60.0,  lp32 = LN2/14.2;
    double t = 0.0;
    double xp = 30.0, yp = 40.0, xl = 240.0, yl = 150.0;
    int i;

    for (i = 0; i < MAXN; ++i, t += 1.0) {
        time[i] = t;
        ca45[i] = exp(-lca45 * t);
        s35[i]  = exp(-ls35  * t);
        i125[i] = exp(-li125 * t);
        p32[i]  = exp(-lp32  * t);
    };

    init();
    scale(time,MAXN,12);
    ca45[MAXN] = 0.0;          /* Startwert                    */
    ca45[MAXN+1] = 0.1;        /* Schrittweite                 */
    ca45[MAXN+2] = 10.0;       /* Anzahl Achsenmarkierungen    */
    for (i = MAXN; i < MAXN+3; ++i) {
        s35[i] = ca45[i];
        i125[i] = ca45[i];
        p32[i] = ca45[i];
    };
    axis(xp,yp,xl,time[MAXN],time[MAXN+1],time[MAXN+2],0,0,"Zeit [Tage]");
    axis(xp,yp,yl,ca45[MAXN],ca45[MAXN+1],ca45[MAXN+2],1,1,
                                                    "rel. Aktivitaet");
    raster(xp,yp,xl,yl,time[MAXN+2],ca45[MAXN+2]);
    line(time,ca45,MAXN,xp,yp,xl,yl,0);
    print(" 45Ca");
    line(time,s35,MAXN,xp,yp,xl,yl,0);
    print(" 35S");
    line(time,i125,MAXN,xp,yp,xl,yl,0);
    print(" 125I");
    line(time,p32,MAXN,xp,yp,xl,yl,0);
    print(" 32P");
    move(xp-10.0,yp-25.0);
    rotate(0);
    print("Abfall der Radioaktivitaet einiger Isotope");
    move(xp-20.0,yp-35.0);
    draw(xp-20.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp+yl+10.0);
    draw(xp+xl+10.0,yp-35.0);
    draw(xp-20.0,yp-35.0);
}
```

Bild. 6. Bei diesem Beispiel wurde die automatische Skalierung durch eine manuelle Skalierung ersetzt

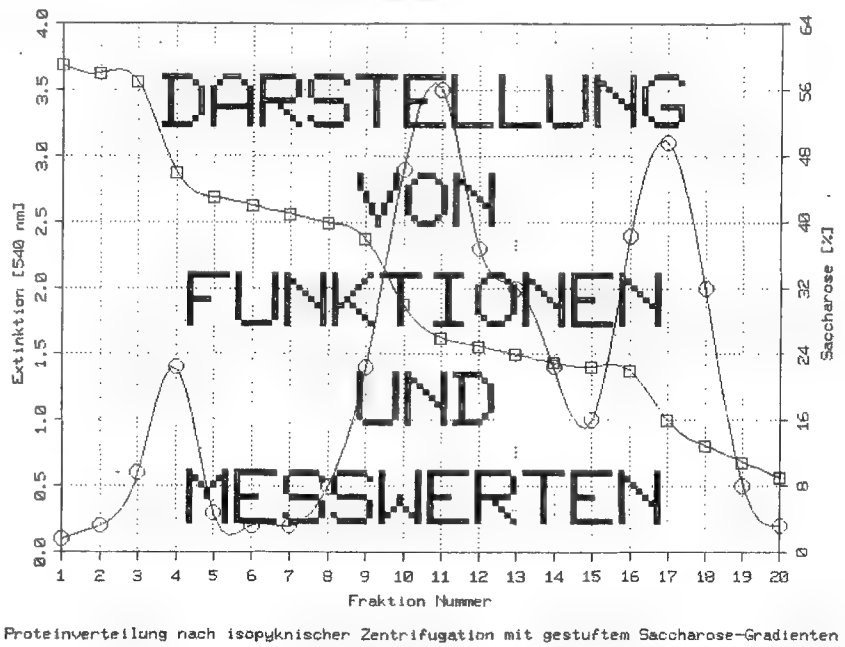


Bild 7. Die Darstellung von Meßwerten und Funktionen ist mit scale.c universell

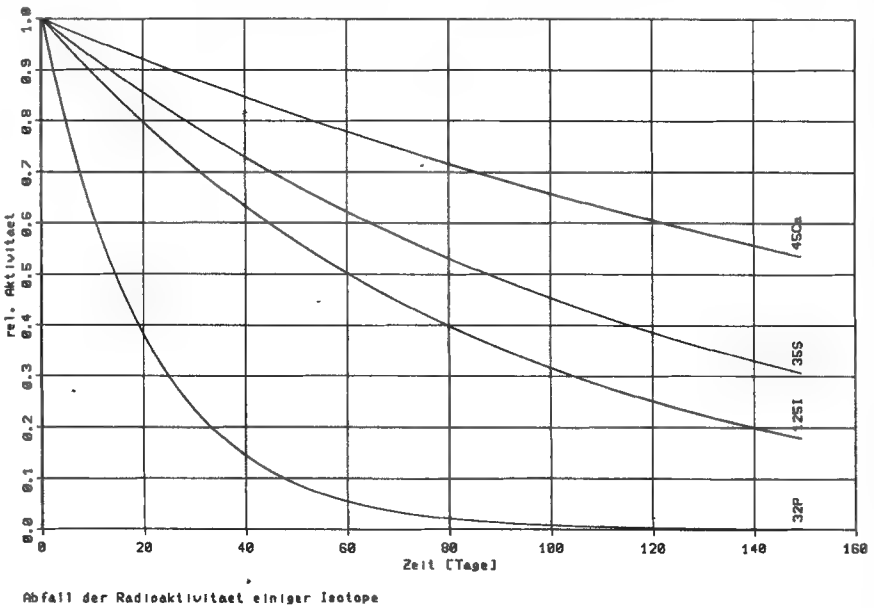


Bild 8. Die Ausgabe des Beispiels aus Bild 6 auf dem Plotter

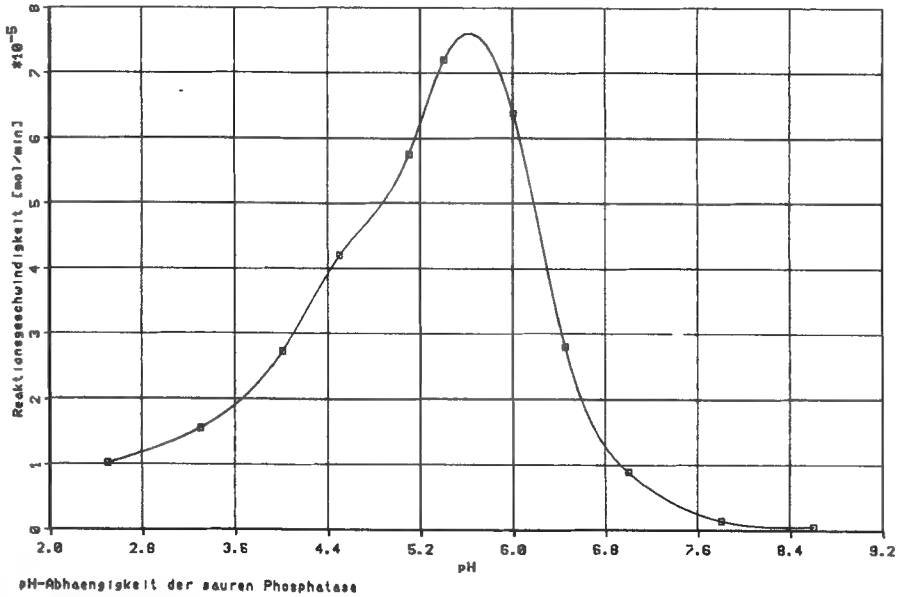


Bild 9. Die Ausgabe des Beispiels aus Bild 4 auf dem Plotter

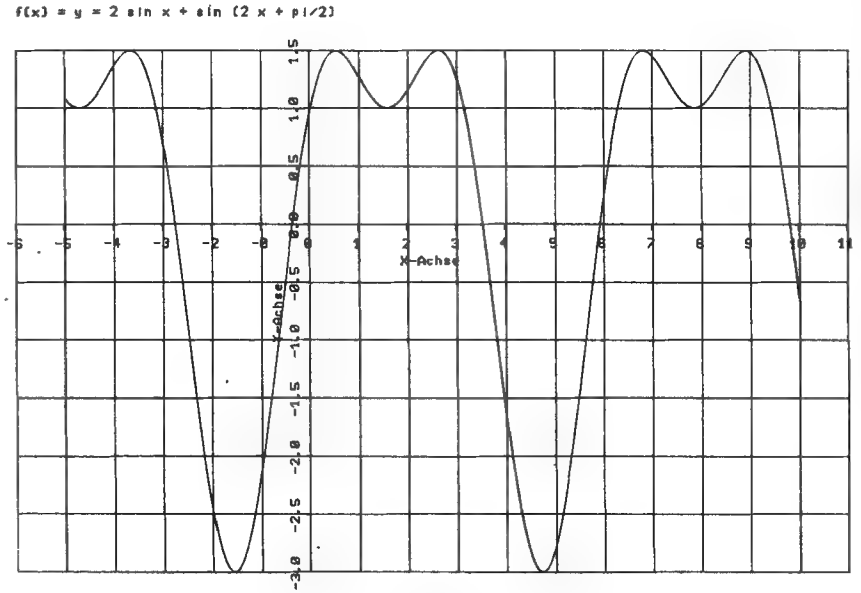


Bild 10. Die Ausgabe des Beispiels aus Bild 5 auf dem Plotter

tion AXIS() zur richtigen Plazierung der Achsenbeschriftung gebraucht. Außerdem werden sechs weitere Funktionen benötigt:

init():	Initialisiert das entsprechende Ausgabegerät.
move():	Fahren mit gehobenem Stift.
draw():	Fahren mit gesenktem Stift.
print():	Gibt einen Text aus.
rotate():	Legt die Schreibrichtung fest. Dabei gilt 0 = waagerecht, rechts 1 = senkrecht, aufwärts
color():	Legt eine Farbe fest. Dabei gilt 0 = Grundfarbe 2 = Farbe für das Raster
symbol():	Zeichnet ein Symbol an die aktuelle Stiftposition.

Die Koordinaten werden grundsätzlich als DOUBLE-Werte übergeben und enthalten Angaben in Millimetern.

Die Auflistung der Funktionen zeigt, was ein Grafik-Ausgabegerät mindestens leisten muß, wenn es zusammen mit „scale.c" eingesetzt werden soll.

Die Datei „gplot.c" (*Bild 2*) zeigt beispielhaft, wie ein Plotter, hier vom Typ ITOH, angekoppelt wird. Der Plotter ist an den Druckerport angeschlossen. Mit XFAK und YFAK werden die Koordinatenangaben so umgerechnet, daß die Ausgaben in Millimeter herauskommen.

Die Schrittweite des hier verwendeten Plotters beträgt 0,1 mm. Die Funktion SYMBOL lenkt die ankommenden Werte so um, daß durch 1 ein Kreis, durch 2 ein Quadrat und durch 3 ein Karo gezeichnet wird.

Der mir zur Verfügung stehende Rechner besitzt den Grafikprozessor NEC 7220. Der Prozessor wird über Systemroutinen (g...) angesprochen. Die Datei „gscreen.c" (*Bild 3*) zeigt, wie man das Paket „scale.c" an diese Bildschirmgrafik ankoppeln kann. Wieder dienen XFAK und YFAK der Koordinatenanpassung, wobei der abgebildete Bereich von 768 × 576 Punkten so behandelt wird, als ob er einem DIN-A4-Blatt entspräche. Da der Ursprung in der linken, oberen Ecke liegt, ist YFAK negativ. Durch YOFF wird er in die untere, linke Ecke verlegt. Die Grafik-Ausgabe ist nur monochrom, deshalb werden verschiedene Farben durch unterschiedliche Linientypen ersetzt; das Raster wird dann gepunktet. Dem Zeichnen von Symbolen wird etwas nachgeholfen, da der NEC 7220 nur Kreise und Rechtecke kennt. Wichtig ist, daß nach dem Zeichnen eines Symbols der Grafik-Cursor wieder an der alten Stelle steht. Wieder entspricht die 1 dem Kreis, die 2 dem Quadrat und die 3 dem Karo.

Spruch des Monats

„Ich bin immer noch gegen Apfelmännchen in mc; ein Apfelweibchen wäre mir lieber."

Aus einem Gespräch unter mc-Redakteuren

Durch die beiden Dateien „gplot.c" und „gscreen.c" können auf zwei völlig verschiedenen Ausgabegeräten fast identische Bilder erzeugt werden. Selbst die Tatsache, daß die Schriftgröße unterschiedlich ist, wird von dem Paket „scale.c" gut verdaut.

Mit entsprechenden Treiber-Dateien sollte es möglich sein, auch vollkommen andere Grafik-Peripheriegeräte zu betreiben. Schließlich ist es auch möglich, vollständig selbst geschriebene Routinen einzusetzen.

Einige Beispiele

Nun zum Lohn der ganzen Mühe. Die Datei „beisp1.c" (*Bild 4*) enthält ein Anwendungsbeispiel, an dem die typische Vorgehensweise gezeigt ist, die eingehalten werden muß, wenn man mit dem Paket „scale.c" arbeiten möchte. Es handelt sich um 11 Meßwerte, die gezeichnet und durch die Spline-Funktion verbunden werden sollen. Die Meßwerte selbst werden in den Feldern XS und YS der Einfachheit halber gleich bei der Initialisierung abgelegt. Wichtig ist, daß diese Felder Platz für mindestens drei weitere Werte enthalten, wo durch den Aufruf von SCALE() die Skalierungskoeffizienten abgelegt werden. Durch AXIS() werden die beiden Achsen gezeichnet, wobei diese Funktion die Skalierungskoeffizienten aus den Datenfeldern übermittelt bekommt. Außerdem bestimmen XP und YP den unteren, linken Eckpunkt und XL und YL die jeweilige Achsenlänge. Durch die Funktion RASTER() wird das Millimeterpapier simuliert. Die Funktion LINE() markiert bei ihrem ersten Aufruf nur die Meßpunkte durch Quadrate (SYMB = −2). Mit dem Aufruf von SPLINE() wird die Spline-Funktion berechnet und in den Feldern XF und YF abgelegt. Auch diese beiden Felder müssen noch Platz für drei weitere Werte enthalten, die aus den Feldern XS und YS kopiert werden.

Diese Voraussetzung muß erfüllt sein, bevor die Funktion LINE() erneut aufgerufen wird und jetzt nur die Spline-Funktion zeichnet (SYMB = 0). Schließlich erhält die Grafik noch einen Titel und, wenn es gefällt, einen Rahmen. Die Ausgabe dieses Programms auf den Plotter und auf den Grafik-Bildschirm (als Kopie mit dem Matrixdrukker) ist in *Bild 9* zu sehen.

Die Datei „beisp2.c" (*Bild 5*) zeichnet eine mathematische Funktion und demonstriert, wie man die Achsen so verschieben kann, daß sie sich im Nullpunkt schneiden (*Bild 10*).

Das dritte Beispiel in der Datei „beisp3.c" (*Bild 6*) ist nicht nur aktuell, sondern zeigt, wie man die Autoskalierung durch manuelle Skalierung ersetzen kann. Da die Y-Achse definitionsgemäß von 0 bis 1 reicht, werden diese Werte „von Hand" eingesetzt.

(Die Funktion SCALE() erzeugt hier übrigens eine Achse, die von 0 bis 1.2 reicht.) Außerdem ist hier (*Bild 8*) wie schon bei dem ersten Beispiel zu sehen, daß, wenn man mehrere Kurven in eine Grafik setzen möchte, die Skalierungskoeffizienten in die jeweiligen Datenfelder umkopiert werden müssen.

Literatur

[1] *Kernighan, B. W. & Ritchie, D. M.*: Programmieren in C. Carl Hanser Verlag, 1983.

[2] *Schwarz, H. R.*: Numerische Mathematik. B. G. Teubner Verlag, 1986.

Peter Talke

RAM-Monitor unter GEM für den Atari ST

Utility für Programmierer

Haben Sie schon mal versucht, sich die Datenbereiche eines GEM-Programmes zu Debugging-Zwecken direkt anzusehen? Suchen Sie ein Hilfsprogramm, mit dem Sie sich Ihre verkorksten GEM-Objekt-Bäume einmal direkt ansehen können? Oder haben Sie schon etliche Stunden damit verbracht, größere Speicherbereiche nach bestimmten Zeichen- oder Bytekombinationen abzusuchen?

Wenn ja, dann sollte zumindest ein Teil Ihrer Probleme durch das Programm Minimon für den Atari ST zu lösen sein. Bei Minimon (in Aktion auf *Bild 1*) handelt es sich um ein in GEM eingebundenes, in C erstelltes Programm, das zur Anzeige, Änderung und Suche nach Speicherinhalten, sowie zum Laden und Abspeichern von Speicherbereichen dient.
Das gewisse Etwas bekommt Minimon durch die ständige Verfügbarkeit unter allen GEM-Anwenderprogrammen, die die bekannte Menüleiste verwenden. Dadurch, daß Minimon als Desk-Accessory programmiert wurde, ist es überall und sofort aktivierbar und erlaubt den Zugriff auf den gesamten, bestückten Speicherbereich des Rechners, einschließlich des I/O-Bereiches. Das Programm Minimon kann auf allen Rechnern der Atari ST-Serie als Desk-Accessory installiert werden. Allerdings ist es nur in der höchsten Grafikauflösung mit Monochrom-Monitor lauffähig. Sonst gibt es keine weiteren Hardwarevoraussetzungen. Minimon muß in das Wurzel-Inhaltsverzeichnis der Boot- bzw. Systemdiskette kopiert werden, von wo es beim Systemstart automatisch installiert wird. Sodann läßt es sich unter dem Titel „DESK" in der Menüleiste durch Anwählen des Eintrages „Mini Monitor" aktivieren. Nach dem Aufruf des RAM-Monitors erscheint das zur Kommunikation verwendete Fenster. In der obersten Zeile des Fensters befindet sich die Kommandozeile mit Eingabecursor. Dort werden die einzelnen Funktionen samt der benötigten Parameter über Schlüsselbuchstaben aufgerufen (Bild 1). Darunter wird der selektierte Speicherbereich als Hexdump und Textäquivalent angezeigt. Da dieser Aufbau festgelegt ist, finden keine Scrollbewegungen statt. Das Fenster kann in der gewohnten Art vergrößert, verkleinert oder auf dem Bildschirm verschoben werden, ebenso kann es gleichzeitig neben anderen Fenstern in der üblichen Weise verwendet werden.
Als ständig abrufbereiter Bedienungshinweis steht ein Hilfsfenster zur Verfügung, auf dem alle erlaubten Kommandos samt benötigter Parameter aufgezeigt werden, falls die Help-Taste betätigt oder ein falsches Kommando eingegeben wurde (*Bild 2*).

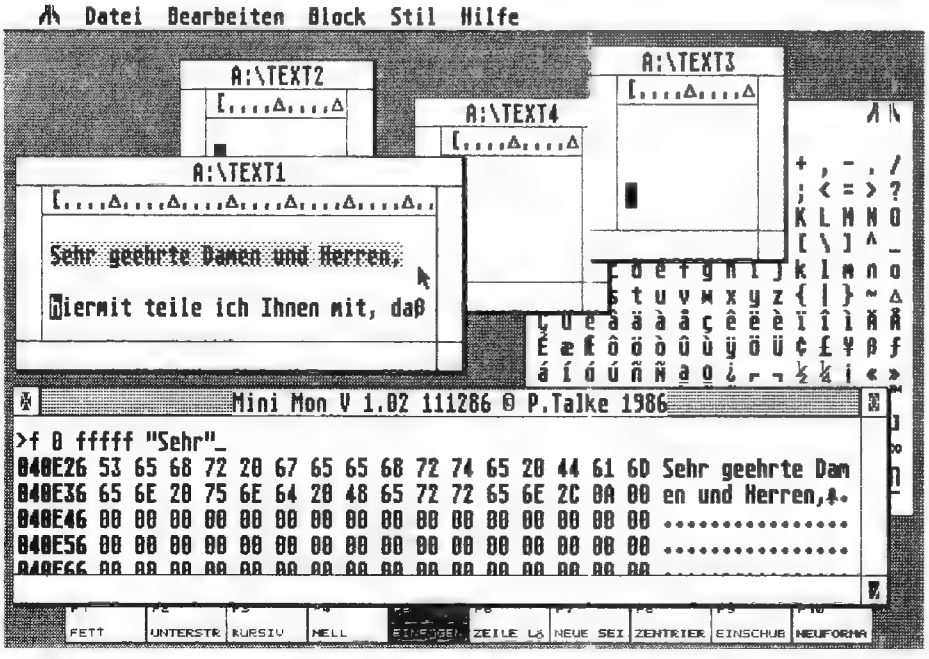


Bild 1. Der RAM-Editor steht jederzeit zur Verfügung

Empfehlungen zur sicheren Bedienung

Als Parameter für Suche und Änderung sind sowohl Hexadezimalzahlen als auch Zeichenketten und Kombinationen aus beiden erlaubt. Alle Adressen oder Speicherinhalte (Strings natürlich ausgenommen) sollten als Hexadezimalzahlen eingegeben werden, falls nicht interessante, aber doch wohl weniger erwünschte Effekte den Bildschirm des Rechners erfüllen sollen. Dabei dürfen Parameter, die Speicherinhalte darstellen sollen, eine Breite von acht Bit nicht überschreiten. Für den Eiligen sei noch hinzugefügt, daß nicht auf unbestückten Adreßraum des Atari ST zugegriffen werden sollte, da dies mit an Sicherheit grenzender Wahrscheinlichkeit zum Absturz des Rechners führen würde. Mehr dazu und zu *Bild 3* aber erst nach Beschreibung der einzelnen Kommandos.

Folgende Kommandos sind erlaubt:

M <start> [<ende>]

Dient zur Anzeige von maximal 256 Bytes ab der angegebenen Startadresse. Als

Default werden 256 Bytes verwendet, falls keine oder eine ungültige Endadresse angegeben wurde.

E <adresse> <byte>|<"text"> [<byte>|<"text">] ...

Änderung der Speicherhinhalte ab der angegebenen Adresse.

F <start> <ende> <byte>|<"text"> [<byte>|<"text">] ...

Suche nach angegebener Byte- oder Textkombination im spezifizierten Speicherbereich. Falls die Suchkombination gefunden wurde, wird eine Speicherseite ab der Adresse des ersten Vorkommens ausgegeben.

S <start> <ende> <"dateiname">

Ablegen des Speicherbereiches auf Festplatte, Ramdisk oder Floppydisk. Im Dateinamen dürfen auch Laufwerksbezeichnungen oder Unterverzeichnisse verwendet werden.

L <start> <"dateiname">

Laden einer Datei ab der angegebenen Adresse. Vor Ausführung des Befehls wird der Benutzer über die tatsächliche Länge der Datei informiert und vor die Entscheidung gestellt, den Ladevorgang zu beginnen oder abzubrechen (*Bild 5*). Desweiteren gibt es noch die Direktkommandos „+" oder „–". Hiermit kann die jeweils nachfolgende oder zurückliegende Speicherseite zur Anzeige gebracht werden, wodurch auch ein zügiges Durchblättern von größeren Speicherbereichen möglich ist. Wie auf dem Atari ST üblich, wird durch Betätigen der Escape-Taste die Eingabezeile gelöscht.

Hätte man nicht doch?

Um es vorwegzunehmen: man hätte. Sicherlich haben Sie sich gefragt, warum der auftretende Bus-Error bei Zugriff auf nicht bestückten Adreßraum nicht abgefangen und dem Benutzer in vernünftiger Form angezeigt wird. Dies hängt ein wenig mit der Architektur des in den Atari-ST-Rechnern verwendeten Prozessors M68000 zusammen. Mehr praxisorientierte Leser können sich hier auch mit Bild 3 zufriedengeben. Dort erfahren Sie, auf welche Bereiche Sie zugreifen können, und auf welche nicht. Vorsicht ist besonders im I/O-Bereich geboten, da dort teilweise nur einzelne Adressen oder alle ungeraden Adressen ausdekodiert sind.

Der M68000 hat die eigentlich gutgemeinte und sehr förderliche Eigenschaft, schon den nächstfolgenden Operanden oder Opcode zu laden, während der vorhergehende Befehl noch bearbeitet wird (Prefetching). Gepaart mit der Fähigkeit des Prozessors, auftretende Fehler verschiedenster Art mit Ausnahmebehandlungen zu verarzten (Exceptions) gibt es jedoch Schwierigkeiten. Zu jeder Exception gehört ein über eine Vektortabelle aufzurufendes Fehlerprogramm. Diese geben aber beim Atari ST, wenn überhaupt etwas, sinnigerweise nur „Bömbchen" mit entflammter Zündschnur aus ...

Geht der M68000 in eine Exception, so legt er neben anderen Informationen auch den Programmzähler mit auf dem Stack ab und beginnt das entsprechende Fehlerprogramm. Nur leider wurde der Prozessor bei Auftreten des Busfehlers unter Umständen während des laufenden Befehlszyklus unterbrochen, so daß durch das Prefetching ein schon auf den nächsten Opcode oder Operanden zeigender Programmzähler auf den Stack gelegt werden kann. Wieweit und ob überhaupt der Programmzähler erhöht wurde, ist dann kaum noch festzustellen. So ist die Rückkehr in den Pro-

2 × ACIA 6850
(Tastatur & MIDI)

$ FF	FC06	MIDI ACIA Data
$ FF	FC04	MIDI ACIA Control
$ FF	FC02	Keyboard ACIA Data
$ FF	FC00	Keyboard ACIA Control

MFP 68901
(Timer, IO, Interrupts)

$ FF	FA2F	USART Data
$ FF	FA2D	Transmitter Status
$ FF	FA2B	Receiver Status
$ FF	FA29	USART Control
$ FF	FA27	Sync Character
$ FF	FA25	Timer D Data
$ FF	FA23	Timer C Data
$ FF	FA21	Timer B Data
$ FF	FA1F	Timer A Data
$ FF	FA1D	Timer C / Timer D Control
$ FF	FA1B	Timer B Control
$ FF	FA19	Timer A Control
$ FF	FA1Z	Vektor Register
$ FF	FA15	Interrupt Mask B
$ FF	FA13	Interrupt Mask A
$ FF	FA11	Interrupt In – Service B
$ FF	FA0F	Interrupt In – Service A
$ FF	FA0D	Interrupt Pending B
$ FF	FA0B	Interrupt Pending A
$ FF	FA09	Interrupt Enable B
$ FF	FA07	Interrupt Enable A
$ FF	FA05	Datenrichtung
$ FF	FA03	Flankenregister
$ FF	FA01	Parallelport

YM-2149
(Soundchip

$ FF	8802	Write Data
$ FF	8800	Read Data / Register Select

DMA/FDC

$ FF	860D	DMA Adresse Lowbyte
$ FF	860B	DMA Adresse Midbyte
$ FF	8609	DMA Adresse Highbyte
$ FF	8606	DMA Mode / Status
$ FF	8604	Disk-Controller / Sector Count
$ FF	8602	Reserviert
$ FF	8600	Reserviert

CRTC

$ FF	8260	Bildauflösung
$ FF	8240	– $ FF 825E Farbpalettenregister 0–15
$ FF	820A	Syncronisationsmodus
$ FF	8209	Video Adress Lowbyte
$ FF	8207	Video Adress Midbyte
$ FF	8205	Video Adress Highbyte
$ FF	8203	Bildschirmspeicher Low Adressbyte
$ FF	8201	Bildschirmspeicher High Adressbyte

MMU
(Memory Management Unit)

$ FF	8001	Speicherkonfiguration

192 K ROM
(System – ROM)
$ FE FFFF
$ FC 0000

128 K ROM
(ROM – Modul)
$ FB FFFF
$ FA 0000

512 K RAM
(Bei entsprechendem Speicherausbau)
$ 0F FFFF
$ 08 0000

512 K RAM
$ 07 FFFF
$ 00 0000

Bild 3. Ein Wegweiser durch den Speicher des Atari ST

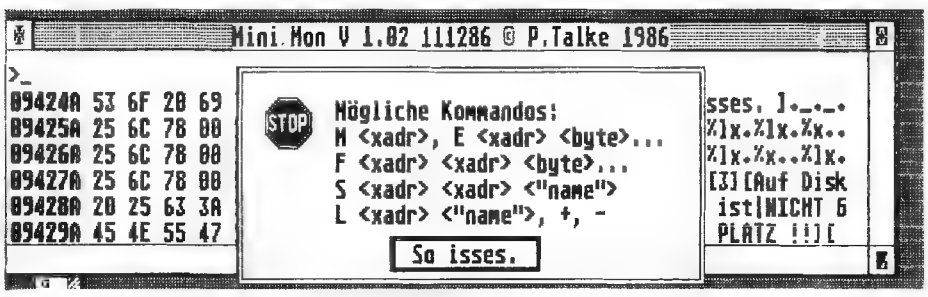


Bild 2. Das Programm gibt Hinweise zur richtigen Bedienung

grammteil, der den Busfehler hervorgerufen hat, etwas erschwert. Dies zufriedenstellend zu lösen wäre in Assembler sicherlich kein Problem, hätte aber den Rahmen dieses C-Programmes gesprengt, weshalb darauf verzichtet wurde.

Desk-Accessory selbst gemacht

Das Programmieren eines „Schreibtisch-Utensils" auf dem Atari ST wurde dem Programmierer zumindest bisher nicht gerade leicht gemacht. Es reichte nicht aus, die verschiedenen Bücher, die zu diesem Thema angeboten werden, durchzuarbeiten. Damit kann höchstens eine Grundkenntnis der GEM-Programmierumgebung vermittelt werden. Einige Punkte auf dem Wege zum Accessory bleiben ungeklärt, andere sind fehlerhaft dokumentiert. Die offizielle Dokumentation des Atari-Entwicklungspaketes ist da keine Ausnahme. Zum Glück hat der frustrierte Programmierer die erfolgversprechende Möglichkeit, bereits vorhandene Accessories zu analysieren, sofern sie in der Quellversion vorliegen. Hier kann und soll Minimon als Beispiel und Anregung dienen.

GEM-Accessories haben die Eigenschaft, daß sie beim Systemstart resident in den Speicher geladen und gestartet werden. Nach der Initialisierung wird praktisch eine einzige übergeordnete Endlosschleife [1] durchlaufen, da Accessory-Programme niemals enden (siehe dazu auch *Bild 4*). Wesentlicher Teil dieser

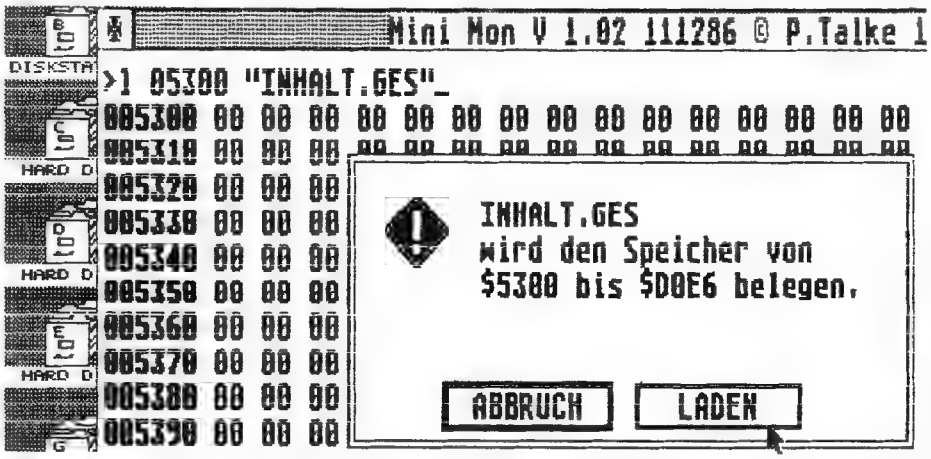


Bild 5. Vor dem Laden eines Programmes meldet sich Minimon zur Sicherheit noch einmal

Schleife sind die GEM-Ereignisfunktionen, speziell die Funktion „Multiple Event". Über diese Funktionen wird unter GEM eine spezielle Art des Multitaskings möglich gemacht. Während das Accessory auf ein vom Programmierer festzulegendes Ereignis oder eine Ereigniskombination, vorzugsweise eine Aktion des Benutzers wartet, wird eine andere Task, z. B. ein weiteres Accessory, oder das gerade laufende Anwenderprogramm bearbeitet. Allerdings gibt es unter GEM keine echten Prioritäten für verschiedene Tasks, und es gibt keine weiteren Umschaltmöglichkeiten zwischen den Tasks. Jede Task ist solange aktiv, solange sie keine weitere Ereignisfunktion aufruft. Erst dann kann das übergeordnete GEM eine andere Task aktivieren.

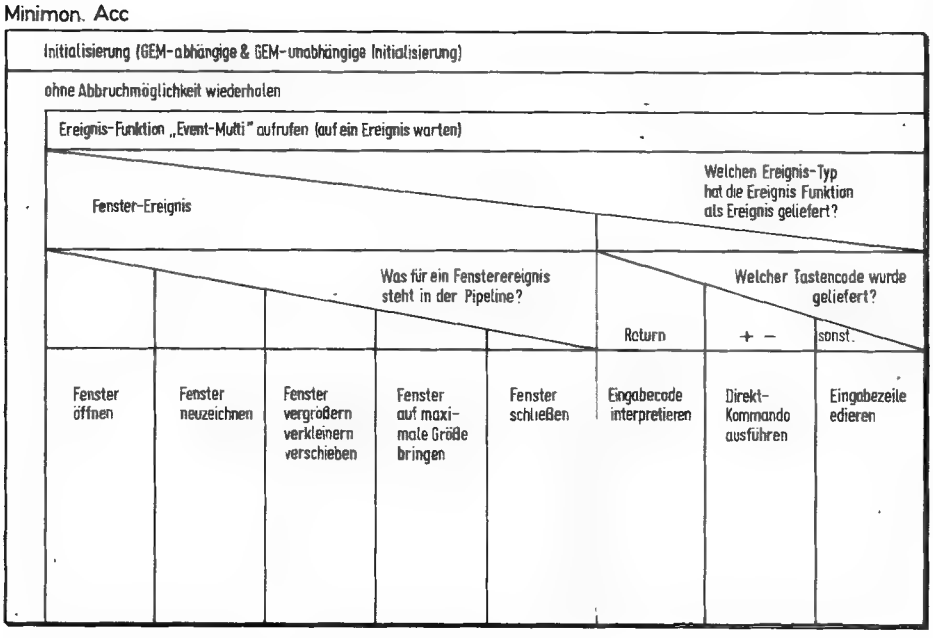


Bild 4. Prinzipieller Aufbau eines Desk Accessories

Pipeline ohne Öl

Für Kommunikationszwecke zwischen GEM und einzelnen Tasks, auch zwischen Tasks, dient ein Puffer, der nach dem Pipeline-Prinzip gelesen oder beschrieben wird. Kern des Minimon-Accessories ist es daher, auf ein auftretendes Ereignis zu warten, die nötigen Parameter, falls erforderlich, aus der Pipeline zu lesen, zu interpretieren und darauf zu reagieren. Nach Abschluß der erforderlichen Aktionen wird auf das nächste Ereignis gewartet.

Die Nachrichten, die aus der Pipeline gelesen werden, haben in der Regel ein festgelegtes Format, es werden Informationen über Typ und Quelle des Ereignisses geliefert sowie wichtige Parameter übermittelt. Allerdings verwenden nicht alle Ereignisfunktionen diese Pipeline, einige verwenden ausschließlich Variablenparameter, andere benutzen gleichzeitig Pipeline und Variablenparameter. Die Funktion „Multiple Event" übergibt auch einige Werte über Variablenparameter, wie z. B. den Code der betätigten Taste.

Die Mehrzahl der auftretenden Ereignisse, die über die Pipeline ausgewertet werden, betreffen jedoch die Fenstertechnik, so auch bei Minimon. Dies ist auch nicht weiter verwunderlich, da sich dem Benutzer doch einige Möglichkeiten bieten, mit Fenstern zu arbeiten. Da muß der Fensterhinhalt des öfteren neu gezeichnet werden, es muß überwacht werden, ob das Fenster verschoben, vergrößert oder geschlossen wurde, ob das Accessory vom Benutzer oder von GEM aktiviert oder deaktiviert wurde. Auch dazu werden dem Programmierer einige leistungsfähige Funktionen zur Verfügung gestellt, hat man ihre

Wirkungsweise und Anwendung erst einmal erfaßt. Dies jedoch im Detail zu beschreiben ist nicht Sinn dieses Artikels, dazu sollten Sie Minimon analysieren und nach eigenen Ideen ändern, sowie die zur GEM-Programmierung angebotene Literatur zu Rate ziehen und die in Minimon vorkommenden GEM-Aufrufe mit den Beschreibungen der Literatur nachvollziehen.

Zu guter Letzt

Jetzt noch ein Hinweis über den nicht GEM-bezogenen Teil von Minimon. Die von der Ereignisfunktion gelieferten Tastencodes werden für drei wesentliche Teile des Programmes verwendet, zur Edierung der Eingabezeile, zum Start der Direktkommandos oder zur Ausführung der Eingabezeile. Bei Ausführung der Eingabezeile werden anhand des eingegebenen Schlüsselbuchstabens die entsprechenden Funktionen aufgerufen, wobei ein Zeiger auf den Rest der Eingabezeile übergeben wird. Die aufgerufene Funktion liest dann nacheinander mit Hilfe der Funktion Getfirst die folgenden Parameter.
Ein Umschreiben auf andere Systeme ist nur für ähnlich aufgebaute GEM-Umgebungen denkbar, was aber Kenntnis der verschiedenen Atari-Spezialitäten und DOS-Funktionen voraussetzt, soweit diese nicht im Deklarations- und Definitionsteil des Quellprogrammes angegeben sind.
Beispielsweise ist eine Anpassung an eine niedrigere Bildschirmauflösung denkbar. Diese könnte auf eine einfachere und auf eine interessantere Weise erfolgen. Die Einfachere wäre, die Größe der auszugebenden Speicherseite so zu verringern, daß sie komplett in ein schirmfüllendes Fenster paßt. Die interessantere Möglichkeit wäre aber, die Fenstertechnik mit ihren Rollbalken und Pfeilknöpfen voll auszuschöpfen, und nur einen variablen Ausschnitt der Speicherseite anzuzeigen.
Der RAM-Editor Minimon wurde mit dem Megamax C-Compiler erstellt. Zum Erstellen einer Accessory ist beim Linken die Datei ACC.L dazuzubinden. Da bei jedem Compiler eine spezielle Startup-Datei für Accessories dazuzubinden ist, sollte bei anderen Compilern die zugehörige Dokumentation zu Rate gezogen werden. Das Programm Minimon ist mit Quellcode über den Franzis-Software-Service zu beziehen.

Literatur

[1] *Peter Wollschlaeger:* mc 3/87, Desk Accessories für Atari ST.

Die CD-ROMs kommen

Compact Disc – Read Only Memory ist die Anwendung der CD-Technologie für den Computer. Diese Technik wird als Nur-Lese-Speicher genutzt. Aufgrund der immensen Speicherkapazität und der Stabilität der gespeicherten Daten bietet das Medium neue Wege:
– zu publizieren und
– zum veröffentlichten Wissen.
CD-ROM ist eine zukunftsträchtige Technologie. Schätzungen zufolge werden bereits 1990 weltweit:
– über 1 Million CD-ROM-Abspielgeräte installiert sein,
– wird das Marktvolumen im Bereich CD-ROM die Vier-Milliarden-DM-Grenze überschritten haben.

Das Speichervolumen von ca. 600 MByte reicht aus, um den Text von über 100 Bibeln miteinander abzulegen. Diese Datenmengen auf der Fläche eines Bierdeckels müssen sinnvoll zugänglich gemacht werden.
Der Lesekopf des CD-ROM-Abspielgerätes braucht maximal zwei Sekunden, um eine Information auf der CD-ROM zu finden. Die Einbindung der etablierten Produktionstechniken der (Audio-)CD erlaubt kostengünstige Vervielfältigung und günstige Prognosen für die Kostenentwicklung der CD-ROM-Abspielgeräte. CD-ROM ist aufgrund dieser Eigenschaften ein ideales Medium für das Archivieren und Verteilen recherchierbarer Informationen wie: Datenbanken, Software, Lexika, Wörterbücher, Nachschlagewerke, Fachpublikationen usw.
Die bisherigen CD-ROM-Produkte entstanden aus der Übernahme von Großrechner-Datenbanken, z. B. Chemical Abstracts/DEC: Health and Safety in Chemistry; oder Batelle/N.A Philips: Compact Video Demo Disc; oder aus Nachschlagewerken, wie z. B. Grolier Encyclopedia und MicroMedex (universelles Nachschlagewerk für Mediziner).
Obwohl ein großer Teil der CD-ROMs in Deutschland gepreßt wird, sind deutsche CD-ROM-Produkte kaum zu haben. Bisher wurden in Deutschland lediglich Demonstrationsprojekte verwirklicht.
Lasec, eine Firma aus dem Berliner Innovations- und Gründerzentrum (BIG) hat eine auf CD-ROM ausgerichtete Volltextdatenbank „Full Search" entwickelt. Durch die Flexibilität des Programms bietet sie maßgeschneiderte Lösungen nach Bedarf und außerdem das komplette Know-how für die Erstellung von CD-ROMs an.
Lasec kann zum Beispiel vier in Zusammenarbeit mit dem Springer-Verlag (Berlin, Heidelberg, New York, London, Paris, Tokio) erstellte Demonstrationsprodukte vorstellen:
Das Zentralblatt für Mathematik, Auszug aus „Medizinische Zentralblätter", Astronomy and Astrophysics Abstracts und Hommel: Handbuch der gefährlichen Güter.
Diese Produkte wurden mit Lasec-Software aufbereitet und zeigen Ausschnitte der Fähigkeiten von Full Search.
Der ergänzende oder alternative Einsatz von CD-ROM zu Publikationszwecken wird immer interessanter. Schon bei niedrigen Auflagen entsprechen die Produktionskosten von CD-ROMs denen des Buchdruckes oder unterschreiten diese sogar. Durch den gleichzeitigen Einsatz von Retrievalsoftware wie Lasec Full Search können darüber hinaus den Anwendern äußerst vorteilhafte Recherchiermöglichkeiten geboten werden.
Charakteristisch für Lasec CD-Produkte ist:
die Datenbasen werden aus für den Satz bestimmten Datenbändern produziert, das Retrievalprogramm arbeitet in deutscher Sprache, mit verständlichen Handlungsanweisungen und eindeutigen Hilfestellungen, Benutzeroberflächen sind in mehreren europäischen Sprachen erzeugbar, die Darstellung der Texte ist dem Druckbild nachempfunden (Proportionalschrift), Grafiken und Bilder können in die Texte integriert werden, mehr als 900 – allen Anforderungen anpaßbare – Zeichen erlauben die Darstellung fach- oder sprachspezifischer Symbole oder Buchstaben sowie komplexer mathematischer Formeln, Volltextindexierung dokumentiert alle Fundstellen jedes beliebigen Wortes blitzschnell und vollständig, die Volltextindexierung kann auch durch eine Schlagwortliste ersetzt werden, Sortierfolgen sind frei definierbar, verknüpfte Suchanfragen mit „und", „oder", „ohne" sind möglich, alle Suchanfragen werden zur Unterstützung der systematischen Recherche protokolliert, eine Verschlüsselung der Datenbasen schützt vor unberechtigtem Zugriff auf Daten.

Nach Unterlagen von Lasec

Harald Lukas-Simonyi

GW-Basic mit DOS-Komfort

Wir stellen Ihnen hier ein GW-Basic-Programm vor, das ein sortiertes Directory ausgibt, auf Tastendruck ausgewählte Dateien auf dem Bildschirm zeigt, Dateien sortiert, löscht, umbenennt, auf verschiedene Disketten kopiert und außerdem noch Wordstar-Texte in ASCII-Dateien umwandelt. Es kann nach Belieben erweitert werden und ist trotzdem in den meisten Funktionen kaum langsamer als ein Assemblerprogramm. Der Shell-Befehl des GW-Basic macht es möglich.

Wer längere Zeit mit seinem Computer gearbeitet hat, steht immer wieder vor dem Problem, ganze Disketteninhalte neu ordnen und bereinigen zu müssen. Einige Dateien sollen gelöscht, einige auf andere Disketten kopiert, wieder andere umbenannt werden; manchmal sind auch Teilkopien erforderlich, ohne daß die MS-DOS-Joker wirklich weiterhelfen. Zwischendurch muß man sich auch die eine oder andere Datei ansehen, um zu wissen, worum es sich überhaupt handelt. Da bleibt dann meist nichts anderes übrig als wiederholtes Eintippen von DOS-Befehlen und Dateinamen.

Diesem Mißstand versucht das Programm SHELLBAS (*Bild*) beizukommen. Es erstellt ein sortiertes Directory und fragt dann Dateiname für Dateiname ab, ob die Datei auf dem Bildschirm gezeigt werden soll. Anschließend stehen fünf Optionen zur Auswahl: Löschen, Kopieren, Umbenennen, Sortieren, Umwandlung von Wordstar-Texten in ASCII-Dateien. Das Programm speichert die gewählte Option und beginnt mit der Ausführung, wenn alle im Directory aufgeführten Dateien abgefragt sind. Grundsätzlich können die Optionen beliebig gemischt werden. Der Ablauf erfolgt automatisch. Nur beim Kopieren, das sich nach allen anderen Optionen vollzieht, muß in der vorliegenden Fassung bestätigt werden, daß die richtige Zieldiskette eingelegt ist.

Um eine solche Vielfalt von Funktionen mit einem relativ kleinen Programm ausführen zu können, findet das Shell-Kommando Verwendung, mit dessen Hilfe sich DOS-Befehle aus einem laufenden Basic-Programm heraus aufrufen lassen. Auf PCs mit 512 oder 640 KByte ist es am zweckmäßigsten, wenn sich auf der RAM-Disk die Dateien GWBASIC.EXE, SORT.EXE, ASSIGN.COM, COMMAND.COM und das Programm selbst befinden; mit einer Batch-Datei ist das Kopieren kein Problem. Die zu bearbeitende Diskette muß sich in Laufwerk b: befinden. Mit dem von c: aus eingegebenen Befehl „assign a:=c:" wird verhindert, daß beim Aufruf von Shell-Kommandos unnötige Zugriffe auf Laufwerk a: erfolgen. Da beim Kopieren alle Dateien auf der RAM-Disk zwischengespeichert werden, muß diese relativ großzügig bemessen werden, zumal die oben genannten Programme rund 100 KByte belegen. Will man bei kleinerem Speicherausbau Platz sparen, empfiehlt es sich, GW-Basic aus Laufwerk b: zu laden. Wenn auch das nicht die nötigen Einsparungen bewirkt, kann man die RAM-Disk den kopierten Dateien vorbehalten und auf den Assign-Befehl verzichten. Jedes Shell-Kommando greift dann auf Laufwerk a: zu, was natürlich die Abläufe verlangsamt.

Zunächst enthält das Programm in den Zeilen 100...180 eine Verteiler-Routine, die spätere Revisionen vereinfacht. Routine 1 zeigt einige Anwendungen des

```
1 'Programm setzt aktuelles Laufwerk c: voraus !
3 'Auf c: Befehl >assign a:=c:< eingeben !
5 'Alle ungeraden Nummern können gelöscht werden.
10 CLS: DEFINT Z: KEY OFF
20 DIM A$(112),B$(112),FLAG(112),Z(255)
30 F$(1)="gelöscht": F$(2)="kopiert": F$(3)="umbenannt"
40 F$(4)="sortiert": F$(5)="konvertiert"
99 '------ VERTEILER -----
100 GOSUB 1000: 'Shell - Directory sortieren
110 GOSUB 1100: 'Dateinamen in String-Feld
120 GOSUB 1200: 'Shell - Dateien zeigen; Abfrage
130 FOR P=0 TO N-1: IF FLAG(P)=0 THEN GOTO 160
140   CLS: PRINT A$(P)+" wird "+F$(FLAG(P))
150   ON FLAG(P) GOSUB 1300,1400,1500,1600,1900
151 'Shell - löschen; kopieren; rename; sortieren; WS-ASCII
160 NEXT
170 FOR P=0 TO N-1: IF FLAG(P)=2 THEN GOSUB 1410
180 NEXT: 'Abschluß des Kopierens
200 CLS: PRINT"- Folgende Dateien wurden bearbeitet:"
210 FOR P=0 TO N-1: IF FLAG(P)=0 THEN GOTO 230
220   PRINT A$(P)+" "+F$(FLAG(P))
230 NEXT: KILL "DIRDAT1": END
999 '------ ROUTINE 1: Sort. Directory -----
1000 SHELL "dir b:*.* >dirdat"
1010 SHELL "sort <dirdat >dirdat1"
1020 SHELL "del dirdat": RETURN
1099 '------ ROUTINE 2: Dateinamen bilden -----
1100 CLS: OPEN"i",1,"dirdat1"
1110 IF EOF(1) THEN CLOSE#1:INPUT" Weiter (RETURN)";D:RETURN
1120 LINE INPUT#1,A$(N): A$=""
1130 IF MID$(A$(N),9,1)<>" " THEN N=N-1: GOTO 1190
1140 FOR R=1 TO 12: L$=MID$(A$(N),R,1)
1150 IF R=9 THEN L$=".": B$(N)=A$+".wst"
1160 IF L$=" " THEN GOTO 1180
1170 A$=A$+L$
1180 NEXT R: A$(N)=A$: PRINT A$(N)
1190 N=N+1: GOTO 1110
1199 '----- ROUTINE 3: Zeigen/Abfragen -----
1200 FOR P=0 TO N-1: CLS: PRINT "------ "+A$(P)
1210 PRINT"- Soll diese Datei gezeigt werden ?": GOSUB 10000
1220 IF A=2 THEN GOTO 1240
1230 A$="type b:"+A$(P): PRINT A$: SHELL A$
```

Das Programm muß unter dem Namen SHELLBAS.BAS abgespeichert werden

Shell-Befehls. Sie erstellt eine Datei mit einem sortierten Directory. Einsortiert sind allerdings nicht nur die Dateien, sondern auch die Kopf- und Fußbemerkungen der üblichen Directory-Ausgabe. Zeile 1130 schließt diese Zusätze später aus. Der als Shell-Kommando aufgerufene Del-Befehl kann natürlich ebensogut durch den entsprechenden Kill-Befehl ersetzt werden.

Routine 2 liest die Dateinamen ein, sondert alle im Directory-Format enthaltenen Leerstellen aus und bildet zwei Namen-Felder. Dabei enthält a$(n) die alte Dateibezeichnung und b$(n) eine Hilfsbezeichnung mit der Dateierweiterung „wst" zur Wordstar-Konvertierung.
Routine 3 fragt nun am oberen Bildschirmrand ab, ob die jeweilige Datei gezeigt werden soll. Das Zeigen erfolgt mit Hilfe des Shell- und des Type-Befehls. Wie auf der DOS-Ebene kann mit Ctrl-Num-Lock die Ausgabe angehalten oder sogar mit Ctrl-Scroll-Lock vorzeitig abgebrochen werden. Die letztere Tastenkombination darf aber nicht im Augenblick der Beendigung der Type-Routine betätigt werden, weil dann das Basic-Programm selbst abbricht.
Am unteren Bildschirmrand erscheint eine Informationszeile, die alle Optionen zeigt. Je nach Option erhält die Variable flag(p) einen Wert zwischen 1 und 5, der für die weitere Programmsteuerung durch die bereits erwähnte Verteiler-Routine dient.
Die Routinen 1...3 leisten die Vorarbeiten für die im weiteren Verlauf folgenden, leicht erweiterbaren oder austauschbaren Routinen. Dank des Shell-Befehls sind einige dieser Programmteile echte Einzeiler, die keines Kommentares bedürfen. Die Kopierroutine hat zwei Teile; der eine kopiert die vorgesehenen Dateien auf die RAM-Disk, der zweite überträgt nach Abschluß aller anderen Optionen diese Dateien auf die gewünschte Diskette.
Routine 8 dient der Umwandlung von Wordstar-Texten in ASCII-Code. Sie kopiert jeweils eine zur Konvertierung vorgesehene Datei unter ihrem Hilfsnamen auf die RAM-Disk, um die Zahl der zeitaufwendigen Diskzugriffe zu begrenzen. Diese Datei wird anschließend umgewandelt und unter ihrem alten Namen auf die RAM-Disk geschrieben. Nach Abschluß der Konvertierung wird die umgewandelte Datei unter ihrem alten Namen auf die Diskette kopiert, wobei wieder der Shell-Befehl Verwendung findet. Um die RAM-Disk zu entlasten, werden die Einträge a$(p) und b$(p) nach Beendigung gelöscht, ehe die nächste Datei umgewandelt wird.
Die in Basic zwangsläufig langsame Subroutine ab Zeile 2000 kann leicht durch eine andere ersetzt werden, wobei nur die Dateinamen entsprechend zu berücksichtigen sind.
Die Hilfsroutine am Programmschluß, die zur Feststellung der Ja/Nein-Entscheidung dient, wirkt unter Umständen etwas befremdend. Sie soll zunächst eine blinde Eingabe ermöglichen, was bei den unmittelbar benachbarten J- und N-Tasten sicher nicht der Fall ist. Außerdem stellt sie im Bedarfsfall für Verzweigungen mit ON...GOTO oder ON...GOSUB das erforderliche Wertepaar bereit.
Da DOS Fehler abfängt, sind nicht ausführbare Funktionen für den Programmablauf unkritisch. Alle Programmblöcke sind mit einem RETURN-Befehl abgeschlossen, was die Austauschbarkeit erhöht. Bei Änderungen sind die allgemeinen Programmvariablen a$(p), flag(p) und n (n ist die Zahl der Dateien) zu beachten. Die Variable b$(p) kann innerhalb einer Routine für alle Dateien, deren flag(p) auf diese Routine zeigt, frei verändert werden, wie dies in der Rename-Routine geschieht.
Zu beachten ist, daß eine Anwendung der Kopier-Option auf „COMMAND.COM" und die übrigen für das einwandfreie Funktionieren benötigten Programme diese auf der RAM-Disk überschreibt. Ein versehentliches Löschen ist dabei dank der Zeilen 1440...1460, wo die Großschreibung der Dateinamen unbedingt einzuhalten ist, nicht möglich. Aber das Kopieren eines geänderten und nicht mehr kompatiblen „COMMAND.COM" könnte Fehler im Programmlauf hervorrufen. „SHELLBAS.BAS" wird als Name des Programmes selbst vorausgesetzt.
Die Flexibilität des Shell-Kommandos erleichtert Anpassungen an eigene Komfortbedürfnisse erheblich. In Routine 1 kann man z. B. leicht statt „dir b:*.*" eine zusätzliche Input-Zeile zur Festlegung des Directory-Strings vorsehen. Ebenso läßt sich auch die Sort-Routine durch Einfügen eines vorherigen Abfragens erweitern.
Natürlich sind auch im Basic-Eingabe-Modus jederzeit DOS-Funktionen mit Shell-Befehlen zugänglich.

```
1240 LOCATE 24,1: PRINT: PRINT "===== "+A$(P)
1250 PRINT"- Art der Bearbeitung ?    "
1260 PRINT"1 = Lösche, 2 = Kopiere, 3 = Rename, 4 = Sortiere, 5 =
WS-ASCII, SPACE = Nichts"
1270 A$=INKEY$: IF A$="" THEN GOTO 1270
1280 IF ASC(A$)<>32 AND ASC(A$)<49 OR ASC(A$)>53 THEN GOTO 1270
1290 A=VAL(A$): FLAG(P)=A: NEXT: RETURN
1299 '----- ROUTINE 4: Löschen -----
1300 KILL "b:"+A$(P): RETURN
1399 '----- ROUTINE 5: Kopieren -----
1400 A$="copy b:"+A$(P): SHELL A$: RETURN
1410 CLS: PRINT A$(P)
1420 INPUT"Zieldiskette für Kopie in b: eingelegt (RETURN) ";D
1430 A$="copy "+A$(P)+" b:":SHELL A$
1440 IF A$(P)="COMMAND.COM" OR A$(P)="GWBASIC.EXE" THEN RETURN
1450 IF A$(P)="SHELLBAS.BAS" OR A$(P)="SORT.EXE" THEN RETURN
1460 IF A$(P)="ASSIGN.COM" THEN RETURN: ELSE KILL A$(P): RETURN
1499 '----- ROUTINE 6: Rename -----
1500 INPUT"Neuer Name ";B$(P)
1510 A$="rename b:"+A$(P)+" "+B$(P): SHELL A$: RETURN
1599 '----- ROUTINE 7: Sortieren -----
1600 A$="sort <b:"+A$(P)+" >b:"+A$(P): SHELL A$: RETURN
1899 '----- ROUTINE 8: WS-Konvert. -----
1900 A$="copy b:"+A$(P)+" c:"+B$(P)
1910 SHELL A$
1920 GOSUB 2000
1930 A$="copy c:"+A$(P)+" b:"
1940 SHELL A$: KILL A$(P): KILL B$(P)
1950 RETURN
1999 '------ SUBROUTINE WS-KONVERT. ------
2000 FOR Q=1 TO 159: Z(Q)=Q: NEXT
2010 FOR Q=161 TO 254: Z(Q)=Q-128: NEXT
2020 Z(160)=32: Z(31)=45: Z(141)=32
2030 OPEN"I",2,B$(P): OPEN"O",3,A$(P)
2040 IF EOF(2) THEN CLOSE 2: CLOSE 3: GOTO 2090
2050 Z$=INPUT$(1,2):Q=ASC(Z$)
2070 C$=CHR$(Z(Q)): PRINT#3,C$;: IF Q=31 THEN PRINT#3," ";
2080 GOTO 2040
2090 RETURN
10000 '------ ALLGEM. HILFSROUTINEN ------
10010 PRINT"----- JA = RETURN-TASTE ===== NEIN = LEERTASTE"
10020 A$=INKEY$:IF A$=CHR$(13) THEN A=1: RETURN
10030 IF A$=CHR$(32) THEN A=2: RETURN
10040 GOTO 10020
```

Stefan und Meinolf Bräutigam

Software für Videodat-300

CPC-664 empfängt Daten aus dem Fernsehbild

Der WDR-Computerclub hat mit Videodat-300 ein Verfahren entwickelt, das Daten in einem Fernsehbild überträgt. Unsere beiden Autoren geben eine Anleitung für die Entwicklung von Software am Beispiel des Schneider CPC-664.

In mc 6/1986 [1] wurde das Datenübertragungsverfahren Videodat 300 einschließlich einer Decoder-Schaltung vorgestellt. Videodat erlaubt es, digitale Daten mit einer Geschwindigkeit von 300 Baud in Fernsehübertragungen einzutasten, ohne die Bild- und Tonqualität wesentlich zu beeinträchtigen. Am Empfangsort decodiert eine einfache Schaltung die im Videosignal eines Fernsehempfängers enthaltenen Daten, so daß diese über Normschnittstellen einem Computer zugeführt werden können. An einem Programm für den Schneider CPC-664 soll gezeigt werden, wie Software für das Videodat-Verfahren entwickelt werden kann. Zur Zeit nutzt der WDR das Verfahren regelmäßig in seinen Sendereihen „Computer Club", „Computer Shop" und der inzwischen jedoch ausgelaufenen Schulfunkreihe „Bit und Byte".
Da in mc 6/1986 und in der WDR-Computer-Club-Praxis [2] ausführliche Beschreibungen des Verfahrens und von Decodern zu finden sind, werden hier Systemparameter und Schaltungseinzelheiten nur genannt, wenn sie Einfluß auf die Softwareentwicklung hatten.

Geblockte Übertragung

Die meisten Heimcomputer sind auch unter Basic in der Lage, bei der von Videodat genutzten Übertragungsgeschwindigkeit (300 Baud) eine serielle oder parallele Schnittstelle zu bedienen, die aufgenommenen Daten in einem Pufferspeicher im Rechner abzulegen und auf dem Bildschirm anzuzeigen. Für ein Sichern der Daten auf Diskette oder Platte bleibt jedoch dabei keine Zeit mehr. Die bei Datenfernübertragungssoftware übliche Synchronisation zwischen Empfänger und Sender, z.B. mit dem Xon-/Xoff-Protokoll, ist hier natürlich nicht möglich. Videodat sieht deshalb eine geblockte Übertragung vor: nachdem ein Block von 2 KByte gesendet wurde, wird eine Pause von 15 Sekunden eingelegt, in der der Rechner die Daten aus dem Puffer auf Diskette oder Platte sichern kann.
Das Programm muß das Blockende in der Übertragung erkennen. Obwohl Videodat die Größe eines Blocks mit 2 KByte festgelegt hat, ist es jedoch nicht empfehlenswert, das Ende eines Blocks aus der Anzahl der empfangenen Zeichen abzuleiten. Wird eine Übertragung gestört und es fällt auch nur ein Zeichen aus, ist die Datenaufnahme für den gesamten Rest der Sendung fehlerhaft, da der Rechner Blockgrenzen stets an den falschen Stellen vermutet. Besser ist es, das Ende eines Blocks an den Übertragungspausen selbst zu erkennen. Hierzu wird nach der Übernahme eines jeden Zeichens aus dem Videodat-Decoder ein Zähler inkrementiert. Das Blockende und der Beginn der Übertragungspause (15 Sekunden) wird daran erkannt, daß der Zählerstand einen bestimmten Wert überschreitet, bevor ein weiteres Byte empfangen wird. Der Rechner darf jetzt die gesammelten Daten auf Diskette oder Platte ablegen. Das System synchronisiert sich bei gestörter Übertragung selbst wieder am nächsten Blockende. Die Zeit, die der Rechner am Blockende vergeblich auf das nächste Byte wartet, verkürzt jedoch die Zeitspanne, die dem Rechner für die Aufbereitung und Speicherung der Daten auf Diskette oder Platte verbleibt. Sie sollte deshalb nicht unnötig lang gewählt werden. Messungen zeigten, daß andererseits auch während der Übertragung innerhalb eines Blocks kürzere Pausen auftreten, die nicht als Blockende interpretiert werden dürfen. In Versuchen haben Wartezeiten zwischen 1 und 2 Sekunden gute Ergebnisse geliefert. Zur Aufbereitung und Abspeicherung der Daten in den Pausen bleiben immer noch 13...14 Sekunden.
Nach den bisherigen Überlegungen reicht im Rechner ein Pufferspeicher von 2 KByte Größe zur Zwischenspeicherung der empfangenen Daten aus. Es empfiehlt sich jedoch, einen möglichst großen Puffer vorzusehen, um auch kürzere, ungeblockte Datenübertragungen aufnehmen zu können. In der Schulfunksendung „Bit und Byte" hielt sich der WDR selbst nicht an seinen eigenen Standard. Die Videodat-Übertragung lief etwa 10 Minuten ungeblockt ohne jede Pause. Die Daten mußten vollständig im Pufferbereich des Rechners abgelegt werden, da erst nach Ablauf der Übertragung Zeit für das Abspeichern auf Diskette oder Platte blieb.

Programmbeschreibung

Das Programm (Bild 1) für den Schneider CPC-664 setzt voraus, daß der Videodat-Decoder mit der seriellen Schnittstelle des Rechners verbunden ist.
Gleich zu Beginn wird in den Zeilen 100 bis 150 ein Pufferspeicher angelegt, der, mit einer gewissen Reserve für die Benutzung durch das Betriebssystem (z.B. für den Diskettenpuffer), den gesamten verbleibenden Speicher nach Laden des Programms nutzt. Im Eingangsdialog (Zeilen 170...330) wird die Größe dieses Puffers angezeigt, der Name der Diskettendatei erfragt und die Datei eröffnet. Die serielle Schnittstelle ist über die Adressen F8E0H bis F8EFH erreichbar und verwendet den Serial Timer Interrupt Controller (STI) MK3801. Die Anweisungen in den Zeilen 400 bis 540 stellen die Schnittstelle entsprechend der Videodat-Übertragungsparameter ein und schalten sie auf Empfang [3]. Im zweiten Teil des Eingangsdialogs (Zeilen 550...560) werden Bedienungsanweisungen gegeben. Der Datenempfang wird mit der Enter-Taste gestartet und mit der Tastenkombination Control-Z abgebrochen. Das erste empfangene Zeichen wird ignoriert. Ein Zeiger wird auf den Pufferanfang gesetzt (Zeile 630) und damit der Puffer initialisiert. In Zeile 640 wird der Zähler für die Blockendeerkennung zurückgesetzt. Ergibt die Abfrage in Zeile 730, daß ein Zeichen empfangen wurde, wird das Zeichen im Pufferspeicher abgelegt und auf dem Bildschirm dargestellt. Ist die obere Puffergrenze noch nicht überschritten, wird

der Zähler rückgesetzt und erneut auf den Empfang eines Zeichens gewartet (Zeile 640). Stellt die Abfrage (Zeile 730) fest, daß kein Zeichen empfangen wurde, wird der Zähler inkrementiert (Zeile 750). Diese Abfrage mit Weiterschalten des Zählers wird solange wiederholt, bis entweder ein Zeichen empfangen wurde oder der Zähler seinen Endstand erreicht hat. Befinden sich jetzt Zeichen im Puffer, werden sie auf Diskette übertragen (Zeile 960). Wurde der Empfang der Daten nicht mit Control-Z abgebrochen, bereitet das Programm den Empfang eines weiteren Datenblocks vor (Zeile 610). Beim Abbruch mit Control-Z wird die Datei geschlossen und der Pufferspeicher freigegeben. Die Daten auf der Diskette sind im ASCII-Format und von CP/M und Basic aus zu lesen.

Durch Aktivieren der Zeile 930 läßt sich der Schwellwert des Zeitzählers einstellen. Empfängt der Rechner keine Daten, versucht das Programm nach Erreichen des Zählerendstandes, die Daten auf Diskette zu sichern und signalisiert dies „jetzt" durch einen Piepton. Die Intervalle entsprechen genau der Wartezeit in der Videodat-Blockendeerkennung. Eine Konstante (t0%) in Zeile 760 bestimmt die Länge der Wartezeit.

So wird das Programm bedient

Nachdem alle Geräte (Fernseher, Decoder, Schnittstelle, Rechner) verbunden und betriebsbereit sind, startet man das Programm. Erscheint in der linken oberen Ecke des Fernsehbildes das Videodat-Fenster, wird der Empfang mit der Enter-Taste gestartet. Empfangene Daten können auf dem Rechnerbildschirm mitgelesen werden. Am Ende der Videodat-Sendung sollte, noch bevor das schwarze Fenster in der linken oberen Ecke des Fernsehbildes verschwindet, die Aufzeichnung mit Control-Z abgebrochen werden. Ein Abbruch des Programms über Escape schließt nicht die Datei und gibt auch den Pufferspeicher nicht wieder für das Basic frei.

Die manuelle Bedienung des Aufzeichnungsprogrammms unter Beobachtung des Fernsehbildes wäre nicht nötig, wenn Videodat jede Sendung mit einer genau festgelegten Zeichenfolge beginnen und mit einer weiteren genau festgelegten Folge schließen würde. Solche Zeichenfolgen könnten vom Rechner erkannt und zum selbständigen Start und Abbruch der Datenaufzeichnung verwendet werden. Voraussetzung ist, daß die gleichen Zeichenfolgen in den Daten der Übertragung selbst nie auftreten. Geeignete Folgen lassen sich aber leicht finden, da durchaus auch nichtdruckbare Steuerzeichen aus dem ASCII-Vorrat verwendet werden dürfen. Videodat befindet sich aber noch in der Erprobungsphase, so daß sich solche Vereinbarungen noch leicht in den Standard einfügen lassen.

Literatur

[1] *Rohde, Ulrich:* WDR-Computerclub innovativ. mc 1986, Heft 6, Seite 66...67.

[2] WDR-Computerclub Praxis, Ausgabe 4. Postfach, 5000 Köln 100

[3] Schneider Computer Division: Hardwarebeschreibung der seriellen Schnittstelle Schneider RS 232


```
10 '************************************
20 '*   VIDEODAT - Datenerfassung      *
30 '*                                  *
40 '*      von S.+ M. Braeutigam       *
50 '*         (November 1986)          *
60 '************************************
100 '
110 'Puffer reservieren
120 '
130 high=HIMEM 'Obere Speichergrenze
140 buf=high-FRE("")+5000 'Pufferanfang
150 MEMORY buf 'Obere Grenze fuer BASIC
160 MODE 2
170 PRINT:PRINT:PRINT
180 PRINT "                    VIDEODAT - Datenerfassung"
190 PRINT "                    ========================="
195 PRINT:PRINT
200 PRINT "                       Puffer:      ";high-HIMEM
210 PRINT:PRINT:PRINT:PRINT
220 INPUT "  Dateiname:    ",dnam$
230 IF INSTR (1,dnam$,".")=0 THEN dnam$=dnam$+".dat"
300 '
310 'Datei eroeffnen
320 '
330 OPENOUT dnam$
400 '
410 'RS 232 Interface vorbereiten
420 '
430 OUT &F8E8,1 'Timer data
440 OUT &F8E0,&1A
450 OUT &F8E8,2
460 OUT &F8E0,&1A
470 OUT &F8E8,6 'Data direction
480 OUT &F8E0,3
490 OUT &F8E8,7 'Timer status
500 OUT &F8E0,&33
510 OUT &F8EC,&88 'Control
520 rsr%=&F8ED
530 rdb%=&F8EF
540 OUT rsr%,1 'Receiver enable
550 PRINT:PRINT:PRINT "  Abbruch  mit  <CTRL>Z"
560 PRINT:PRINT:INPUT "  Start jetzt mit [ENTER]",x$:CLS
570 T= INP(rdb%) 'Data Register leeren
600 '
610 'Block lesen
620 '
630 bp=buf 'Pufferzeiger setzen
640 t0%=0 'Wartezeitzaehler zuruecksetzen
700 '
710 'Warten auf Zeichen
720 '
730 IF INP(rsr%)>=128 GOTO 830 'Zeichen
750 t0%=t0%+1
760 IF t0%<400 GOTO 730 'ca. 1.8 sec.
770 GOTO 910
800 '
810 'Zeichen ablegen
820 '
830 bp=bp+1
840 POKE bp,INP(rdb%) 'Zeichen in Puffer
850 PRINT CHR$(PEEK(bp)); 'Zeichen auf Bildschirm
860 IF bp<high GOTO 640 'Puffer-Ueberlauf pruefen
900 '
910 'Puffer schreiben
920 '
930 'PRINT CHR$(7);
940 IF bp=buf GOTO 980 'keine Zeichen im Puffer
950 PRINT CHR$(7);
960 FOR n=buf+1 TO bp:PRINT#9,CHR$(PEEK(n));:NEXT 'auf Diskette
970 PRINT CHR$(7);
980 IF INKEY$<>CHR$(26) GOTO 610
1000 '
1010 'Abbruch
1020 '
1030 CLOSEOUT:MEMORY high
1040 STOP
```

Bild 1.
Der Schneider CPC-664 empfängt Videodat

Udo Kemle

Abfangen von Laufwerksfehlern unter CP/M

Turbo-Pascal prüft, ob auf ein ausgewähltes Laufwerk zugegriffen werden kann

Leider werden mit der Compileroption $I-, die I/O-Fehler durch die Standardfunktion IORESULT prüft, im Zusammenhang mit Diskettenoperationen nicht alle Fehler abgefangen.

Wählt man mit der ASSIGN-Anweisung eine Datei auf einem im System nicht installierten Laufwerk aus, so steigt das Programm bei einem nachfolgenden Diskettenzugriff (z. B. mit RESET) mit der Meldung "BDOS Err On X: Select" aus und man findet sich auf Betriebssystemebene wieder. Hierzu ein Beispiel:

```
LISTING: DRIVE.EXT

(*************************************************************************)
(*                                                                       *)
(*      Die Funktion TEST_DRIVE prueft, ob das Laufwerk                  *)
(*                    verfuegbar ist.                                    *)
(*                                                                       *)
(*  Die Funktion gibt folgende Werte zurueck:                            *)
(*                                                                       *)
(*  0 = oK                                                               *)
(*  1 = Laufwerk ist dem System nicht bekannt                            *)
(*  2 = Laufwerk ist nicht verfuegbar                                    *)
(*      (z.B. keine Diskette im Laufwerk)                                *)
(*                                                                       *)
(*************************************************************************)

function test_drive (drive : char) : byte;

var
   drive_nr,
   current_drive : integer;
   dma           : array [0..127] of char;

begin

   bdos($1a,addr(dma));                 {DMA Adresse setzen            }
   current_drive := bdos($19);          {aktuelles Laufwerk merken     }
   if drive <> ' '
   then begin
      drive    := upcase(drive);
      drive_nr := ord(drive) - 65;
      if bioshl(8,drive_nr) = 0         {neues Laufwerk setzen         }
      then begin                        {wenn <> 0, ist das Laufwerk   }
         test_drive := 1;               {dem System nicht bekannt      }
         bios(8,current_drive);         {ursprungliches Laufwerk       }
         exit;                          {setzen                        }
      end;
   end;
   bios(7);                             {auf Track 00 aufsetzen        }
   if bios(12) <> 0                     {Sektor lesen, wenn Return-Code}
   then                                 {<> 0, ist Lesen nicht moeglich}
      test_drive := 2
   else
      test_drive := 0;
   bios(8,current_drive);
end;
```

Bild 1. Dieses Turbo-Pascal-Programm prüft die Diskettenlaufwerke mit BIOS-Funktionen

Auf einem Computersystem mit zwei Diskettenlaufwerken (A und B) wird folgendes Programm gestartet:

```
var f : file;
begin
  (*$i-*)
  assign(f,'c:xyz'); (* Laufwerk C: *)
  reset(f);
end.
```

Die Folge ist:
BDOS Err On C: Select
A)

Außer der unerlaubten Zuweisung eines Laufwerks gibt es noch einen zweiten Fehler.

Greift man auf eine Diskette zu, hat aber vergessen den Diskettenschacht zu schließen, wird einem die Meldung "BDOS Err On X: Bad Sector" präsentiert.

Bei einem Lesezugriff kann man nach dem Schließen des Diskettenschachtes mit RETURN fortfahren. In maskenorientierten Programmen stört die Fehlermeldung.

Bei einem Schreibzugriff wird jedoch nach RETURN durch "BDOS Err On X: R/O" das Programm beendet.
Die Fehlermeldungen resultieren aus der Tatsache, daß Turbo-Pascal für Diskettenoperationen die BDOS-Schnittstelle verwendet.

Entscheidet man sich dagegen für BIOS-Funktionen, so erhält man Return-Codes zurück und kann somit im Fehlerfall darauf reagieren.

BIOS statt BDOS

Ist das zu prüfende Laufwerk nicht das aktuelle, so wird es mit der BIOS-Funktion 8 (Laufwerk selektieren) ausgewählt.

```
(*****************************************************************)
(*              Demonstration fuer TEST_DRIVE                    *)
(*****************************************************************)

{$idrive.ext}

var
   drive : char;

begin
   write('Laufwerk: ');
   readln(drive);
   case test_drive(drive) of
      0 : writeln('Laufwerk in Ordnung');
      1 : writeln('Laufwerk nicht bekannt');
      2 : writeln('Laufwerk nicht verfuegbar');
   end;
end.
```

Bild 2. So kann die Routine aus Bild 1 eingesetzt werden

Als Rückgabewert erhält man über Register HL die Adresse des „disk parameter headers". Falls der Wert Null zurückgegeben wird, ist das Laufwerk dem System nicht bekannt.

Sollte dies nicht der Fall sein, wird ein Leseversuch auf Spur 0 mit der BIOS-Funktion 12 (Sektor lesen) unternommen, die den Status über Register A mitteilt. Ein Wert ungleich Null bedeutet, daß ein Lesefehler aufgetreten ist (z. B. keine Diskette im Laufwerk).

Die Funktion TEST_DRIVE (*Bild 1*) ist wegen der EXIT-Anweisung nur unter Turbo-Pascal 3.0 lauffähig, EXIT kann auch durch ein GOTO zum Programmende ersetzt werden.

Der Aufruf erfolgt folgendermaßen:

STATUS := TEST_DRIVE(DRIVE);
Dabei ist STATUS eine Variable vom Typ Integer oder Byte, die nach dem Aufruf den Return-Code enthält (Erklärung am Anfang der Funktion).

DRIVE ist eine Konstante oder Variable vom Typ Char, die das zu testende Laufwerk angibt (' ' = aktuelles Laufwerk, 'a' = Drive A:, 'b' = Drive B:, usw.).

Das kleine Beispielprogramm (*Bild 2*) veranschaulicht den Einsatz der Funktion TEST_DRIVE.

Literatur

Bernd Pol: Vom Umgang mit CP/M, CP/M für die Praxis 1, IWT-Verlag.

Preiswerte Software für den PC

PC-SIG (PC Software Interest Group), ein Unternehmen mit Sitz im sonnigen Kalifornien, ist weltweit einer der bedeutendsten Verleger von Software für den PC. Über 700 äußerst preiswerte Disketten mit „Futter" für den PC werden derzeit vertrieben. Die von PC-SIG erhältliche Software ist in zwei Gruppen aufgeteilt: Public Domain Software und User Supported Software.

Was bedeutet Public Domain?

Public Domain Programme sind frei von Urheberrechten. Es ist bei ihnen gestattet, die Software zu kopieren, zu verändern und auch weiterzugeben. Public Domain Software wurde von Autoren geschaffen, die keine formalen Rechte und Lizenzgebühren beanspruchen. Verboten ist es jedoch, Public Domain Programme mit seinem eigenen Namen zu kennzeichnen und diese Programme kommerziell zu vertreiben.

Was bedeutet User Supported Software?

Die Autoren der User Supported Software haben sich entschieden, den kommerziellen Weg zu umgehen, um ihre Programme im direkten Kontakt mit dem Anwender zu vermarkten (Shareware-Konzept). Jede Diskette von PC-SIG mit derartiger Software enthält lauffähige Programme als .COM – oder .EXE-Datei. Wer ein Programm häufig nutzt, sollte dem Autor eine Anerkennung zukommen lassen, worauf dieser in den meisten Fällen dem Anwender das vollständige Handbuch schickt und eventuell auch einen Update-Service vornimmt. Da PC-SIG die Programme der Autoren vor Veröffentlichung prüft, dürften sich unter PC-SIG-Autoren keine „schwarzen Schafe" befinden.

Bei der deutschen Vertretung der PC-SIG ist ein Katalog mit dem Verzeichnis der Programme erhältlich. Einige Bestseller sind darin kurz beschrieben.

15 000 Dateien auf einer CD-ROM!

Um uns in der Redaktion einen Überblick über die Programme zu verschaffen, haben wir uns das neueste von PC-SIG, eine CD-ROM mit einer Auflistung aller Programme und ihrer jeweilig lauffähigen Version, vorführen lassen. Auf dieser CD-ROM mit 550 MByte Speicherkapazität sind der Inhalt von 705 Disketten und Teile der Bibel (in Englisch) untergebracht. Der Inhalt der 15000 Dateien entspricht ungefähr 250 000 Seiten. Damit man diese neueste Speichertechnologie nutzen kann, muß der PC mit einer speziellen Einsteckkarte zum Anschließen des CD-ROM-Laufwerkes ausgerüstet sein. Derzeit ist die Anschaffung eines CD-ROM-Laufwerkes und der entsprechenden Einsteckkarte noch nicht gerade billig, doch werden sich auch hier die Preise nach unten bewegen.

Die CD-ROM wird wie eine Festplatte bedient. Mit dem MS-DOS-Kommando DIR wird das Directory ausgegeben, mit CD das Directory gewechselt und mit COPY Dateien auf Disketten kopiert. Vor dem Gebrauch des CHKDSK-Kommandos (CHKDSK = Check Disk) wird im Handbuch gewarnt – das System stürzt dabei ab.

Interessant ist die PC-SIG-Programmbibliothek für alle MS-DOS-Anwender. Auf einer Diskette mit DOS-Dienstprogrammen befinden sich nützliche Dinge wie z. B. Bedienungshilfen bei DOS-Kommandos, ein Programm zum Anfahren der Parkspur bei Festplatten und ein intelligentes DIR-Kommando.

Nach Unterlagen von Kirschbaum Software, Emmering

Dieter Strauß

SAL: Ein Software-Entwicklungswerkzeug

Mit dem PC-Assembler interaktiv und strukturiert programmieren

Die Programmentwicklung läuft stets nach dem gleichen Schema ab. Nachdem das Programm entworfen ist, wird es codiert und je nachdem ob das Programm in einer Hochsprache oder in Assembler geschrieben ist, wird es compiliert oder assembliert. Da in der Entwicklungsphase zahlreiche Fehler auftreten, muß der Zyklus des Editierens, Compilierens und Linkens unzählige Male durchlaufen werden. Lauer & Wallwitz, ein Softwarehaus mit Sitz in Wiesbaden, hat die Nöte der Programmierer erkannt und ein Steuerprogramm entwickelt, das alle für die Programmentwicklung notwendigen Programme unter „einen Hut bringt".

Das Steuerprogramm SAL (Structured and Assembly Language) schafft eine einheitliche Benutzeroberfläche, in der fast alles integriert ist, was zur Programmentwicklung notwendig ist. SAL integriert einen Editor mit interaktiven Schnittstellen zu Assembler, Compiler, Linker, Debugger und Library Manager und mit einer DOS-Schnittstelle. Damit keine Mißverständnisse entstehen: der Microsoft Assembler (MASM) wird nicht mitgeliefert.
Das Programm wird über Pull-Down-Menüs bedient. Wem dies zu langsam ist, der kann die einzelnen Menüpunkte nach Turbo-Pascal-Art durch Eingabe der hervorgehobenen Buchstaben direkt aufrufen.
Die Benutzeroberfläche von SAL besteht aus zehn Menüs. *Bild 1* zeigt das Menü Setup, das zahlreiche Konfigurationen zuläßt.
Bevor SAL zum ersten Mal aufgerufen wird, muß das Konfigurationsprogramm CONFIGSA.EXE gestartet werden. Auch das Konfigurationsprogramm ist vollständig menügeführt. CONFIGSA.EXE legt eine Konfigurationsdatei an, die auf den Dateinamen SAL.SYS voreingestellt ist. In der DOS-Konfigurationsdatei CONFIG.SYS muß der ANSI-Treiber (DEVICE=ansi.sys) eingebunden sein, damit das SAL-System funktioniert.
Wir haben das SAL-System auf dem MCI XT16SLC, einem PC mit der 8088-CPU (4,77 MHz), der mit zwei Floppylaufwerken ausgestattet ist, getestet. Wer effektiv mit SAL arbeiten will, sollte sich jedoch eine Festplatte zulegen.
Das SAL-System besteht aus einer System-Diskette und dem deutschen Handbuch.

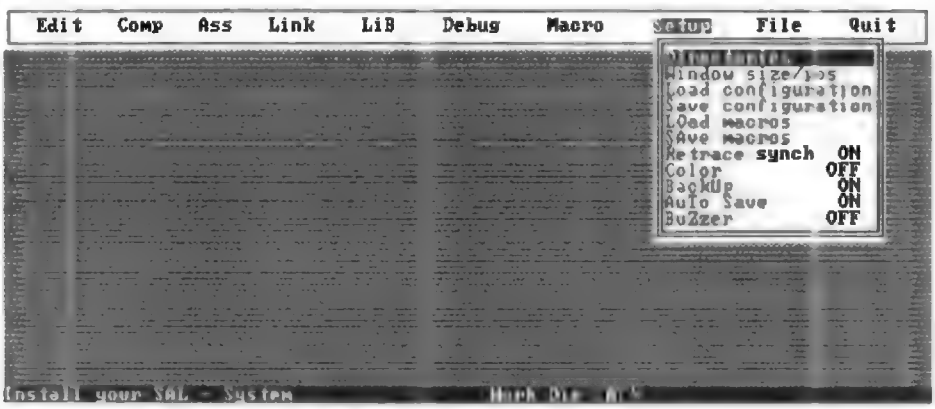


Bild 1. Ein Blick auf das Setup-Menü

Hochsprachenähnliche Programmierung in Assembler

Außer dem Ziel eine einheitliche Entwicklungsoberfläche für die verschiedenen Umgebungen zu schaffen, stand bei der Entwicklung von SAL im Vordergrund, daß die Programmierung in Makro-Assembler einfacher und damit weniger fehleranfällig gestaltet werden soll. Um diesem Traum vieler Programmierer näher zu kommen, wurde in SAL eine Übermenge der Assembler-Sprache – einer Pascal bzw. Modula-2 sehr ähnlichen Sprache – integriert. Strukturierte Blockanweisungen wie IF...THEN...ELSE, WHILE...DO, REPEAT... UNTIL, CASE, FOR...TO...BY...DO müssen unter SAL nicht mehr „zu Fuß" programmiert werden. Gleichzeitig sind alle Assembler-Anweisungen von Microsofts Makro-Assembler (MASM) zugelassen. Damit das SAL-System zwischen MASM-Anweisungen und SAL-Anweisungen unterscheiden kann, müssen alle SAL-Anweisungen in Großbuchstaben geschrieben werden. In unserem ersten Test hatten wir sämtliche SAL-Anweisungen klein geschrieben und wurden dafür mit einer stattlichen Liste an Fehlermeldungen belohnt. Wenn die Gefahr besteht, daß MASM-Anweisungen mit SAL-Anweisungen verwechselt werden könnten, sollten die MASM-Anweisungen in Kleinbuchstaben geschrieben werden. *Bild 2* zeigt die Liste der vom SAL-System reservierten Wörter.
Wie in Hochsprachen üblich, realisiert SAL Unterprogramme mit Prozeduren und Funktionen. Der SAL-Compiler kann Prozeduren und Funktionen so erzeugen, daß ihr Stack und ihre Übergabeparameter mit den Konventionen einer ganzen Reihe von Compilern übereinstimmt. Das Herz aller Pascal-Programmierer wird jetzt höher schlagen,

```
ABS         AH          AL          AND         ARRAY       AT          AUX         AX
BA          BEGIN       BH          BL          BP          BSS         BX          BY
CARRY       CASE        CH          CL          CLRSCR      CODE        COM         COMBINED
COMMON      CONST       CONT        COPYB       COPYW       CS          CURSOR      CX
DATA        DECL        DELLINE     DH          DI          DIV         DL          DO
DOWNTO      DS          DX          ELSE        END         EQUAL       ES          EXE
EXIT        EXPORT      EXTRN       FAR         FILLB       FILLW       FLAGS       FOR
FROM        FUNCTION    HIGH        IDIV        IF          IMOD        IMPORT      INSLINE
INTERRUPT   LARGE       LC3         LENGTH      LOOP        LOW         MAIN        MEDIUM
MEMB        MEMORY      MEMW        MOD         MODULE      MSC3        MSP3        NEAR
NEGATIVE    NOT         OF          OFFSET      OR          OVERFLOW    PAGE        PARA
PARITY      POP         PORTB       PORTW       PROC        PROCEDURE   PTR         PUBLIC
PUSH        QB          READ        READLN      RECORD      RECURSIVE   REENTRANT   REPEAT
RETURN      ROM         SDS         SEG         SEQ         SHORT       SI          SIGN
SIZE        SMALL       SP          SS          STACK       STATIC      STRING      TABLE
THEN        TO          TP3         TYPE        UNTIL       VAR         WHILE       WITH
WRITE       WRITELN     ZERO
```

Bild 2. Das SAL-System kennt zahlreiche reservierte Wörter

```
  Edit   Comp   Ass   Link   LiB   Debug   Macro   Setup   File   Quit
Line:   17   Col: 72   Insert   55047       work.SAL
CASE «id» OF                        ;erzeugt durch Druecken von F10 und C
  «id» :   «stat»
       END
END CASE

FOR «id» := «expr» TO «const» DO    ;erzeugt durch Druecken von F10 und F
  «stat»
END FOR

IF «boolean» THEN                   ;erzeugt durch Druecken von F10 und I
  «stat»
END IF
WHILE «boolean» DO                  ;erzeugt durch Druecken von F10 und W
  «stat»
END WHILE

FUNCTION «id»() : «type»            ;erzeugt durch Druecken von F10 und U
BEGIN
  «stat»
END FUNCTION
```

Bild 3. Tastaturmakros erleichtern das Eintippen

da Turbo-Pascal (Version 3.0) und Microsoft Pascal (3.0) vom SAL-System unterstützt werden. Aber auch die Modula-2- und C-Freunde kommen nicht zu kurz, denn der ITC-M2SDS Modula-2-, der Microsoft C- und der Lattice-C-Compiler werden ebenfalls vom SAL-System erkannt (ab Version 3.0).
Damit der SAL-Compiler weiß, mit welchem Hochsprachen-Compiler er zusammenarbeiten soll, werden die Prozedurköpfe entsprechend gekennzeichnet. In der Liste der reservierten Wörter stehen TP3 für Turbo-Pascal, MSC3 für Microsoft C, LC für Lattice C, MSP für Microsoft Pascal und SDS für ITC M2SDS Modula-2.

Tastaturmakros erhöhen den Bedienungskomfort

Eine weitere wichtige Eigenschaft des SAL-Editors ist die kommandogesteuerte Codegenerierung. Der Editor stellt 24 vordefinierte Funktionstastenmakros zur Verfügung, wobei jedes Makro einen abgeschlossenen Ausschnitt der SAL-Syntax darstellt. Aufgerufen wird ein Tastaturmakro mit der Funktionstaste F10 gefolgt von einem Makrozeichen. *Bild 3* zeigt eine Folge von Anweisungen, die mit Tastaturmakros erzeugt wurden. In spitze Anführungszeichen eingeschlossen sind Platzhalter, welche die an dieser Stelle geforderten Ausdrücke der Syntax symbolisieren. Ein Platzhalter wird einfach mit dem gewünschten Text überschrieben – das fertige Programm wird trotz der komfortablen Bedienung noch nicht auf Knopfdruck erzeugt (!) – und mit der Funktionstaste F9 können Sie den nächsten Platzhalter erreichen.

SAL auf einem Blick

SAL: Structured and Assembly Language

Hardware-Voraussetzungen:

- IBM-PC/AT oder kompatible Computer
- DOS-Version 2.X oder 3.X
- mindestens 384 KByte RAM
- Festplatte empfehlenswert

Interessant für Programmierer, die mit MS-DOS und dem MASM arbeiten

Unter den 24 Tastaturmakros befinden sich so brauchbare Konstrukte wie Repeat...Until, If...Then, Type, String, und Procedure.
Ein kurzes Programm, das fast nichts tut, zeigt der Bildschirmausdruck (*Bild 4*). Im linken Fenster steht der SAL-Quelltext, im rechten der vom SAL-Compiler erzeugte MASM-Quelltext.

Da der SAL-Compiler in der Benutzeroberfläche integriert ist, kann der SAL-Compiler innerhalb des Compile-Menüs oder direkt mit Ctrl KC gestartet werden. Der im Editor stehende Text wird in MASM-Quelltext übersetzt. Das Compilat wird – wie in Bild 4 gezeigt – in das zweite Editorfenster geschrieben, wo es weiter bearbeitet werden kann. Im nächsten Schritt wird meistens der Assembler aufgerufen, der vom Aktivieren des Linkers gefolgt wird. Während dieser ganzen Vorgänge braucht man nicht die SAL-Umgebung zu verlassen. Tippfaule können auch den Menüpunkt Macro aufrufen. Der ganze Vorgang des Compilierens, Assemblierens, Linkens läuft dann automatisch ab. Nur bei Fehlern muß man noch manuell eingreifen.

```
  Edit   Comp   Ass   Link   LiB   Debug   Macro   Setup   File   Quit
Line:   1   Col: 24   Insert   5
MAIN MODULE DEMO 'CODE'
  BEGIN
  END MAIN
END DEMO
                                  Line:   1   Col:  1   Insert   55087
                                  DEMO            SEGMENT PUBLIC 'CODE'
                                                  ASSUME  CS:DEMO
                                  MainDEMO:
                                                  MOV     AH,4CH
                                                  INT     21H
                                  DEMO            ENDS
                                                  END     MainDEMO
```

Bild 4. Ein Bildschirmausdruck mit einem kurzen Programmbeispiel

Komplexe Ausdrücke einfach programmieren

Das SAL-System unterstützt zahlreiche Datentypen. Dazu gehören:
BYTE, WORD, DWORD, ADDR, ARRAY, STRING, RECORD und PROC.

Ähnlich wie in Pascal können auch Aufzählungstypen deklariert werden. Folgende Deklaration ist zulässig:
VAR AmpelStatus : (rot, gruen, gelb).
Mit der TYPE-Anweisung kann der Benutzer eigene Typen definieren.

In Ausdrücken (expressions) sind die arithmetischen Operatoren *, DIV, MOD und der logische Operator AND zugelassen. Komplexe Ausdrücke wie
DX := 4 * CX * AX + I * CX
werden vom SAL-Compiler verstanden und in MASM-Quelltext übersetzt. Innerhalb komplexer Ausdrücke können Register und Variablen zusammen oder getrennt benutzt werden.

Das mitgelieferte Handbuch erläutert auf über 200 Seiten die zahlreichen Möglichkeiten des SAL-Systems. Auch für Programmierer, die mit dem MASM noch nicht gut umgehen können, ist SAL eine Lernhilfe, da sie ein Problem mit den eingebauten Strukturierungshilfen schnell lösen und sich anschließend den erzeugten Assembler-Quelltext anschauen können.

Das SAL-System ist für den Programmierer ein gutes Werkzeug, das zudem preislich sehr interessant ist.

Wettbewerb für Computerkünste in Linz

Neue Trends in der Computerkunst will der 1. Internationale Wettbewerb für Computerkunst, der vom 15. bis 19. September dieses Jahres in Linz stattfindet, aufzeigen. Ziel dieser Präsentation, die mit einer Prämierung der besten Ideen abgeschlossen wird, ist es, den Computer als ein vielseitiges, den schöpferischen Möglichkeiten keine Grenzen setzendes Medium vorzustellen.

Den Wettbewerb hat das ORF-Landesstudio Oberösterreich als Organisator in drei Bereiche gegliedert: In der Sparte Computeranimation werden von einer internationalen Fachjury computergenerierte Animationssequenzen in der Dauer von drei bis fünf Minuten, in der Computergrafik computererzeugte Still Pictures, im dritten Bereich, der Computermusik, schließlich digital produzierte Bänder zwischen acht und zwanzig Minuten Länge bewertet. Dabei sind Kreativität und Originalität der Arbeiten als Hauptkriterien vor den eingesetzten technischen Mitteln ausschlaggebend. Den Siegern winkt der Prix Ars Electronica, der in den Kategorien Computermusik und Computergrafik mit jeweils 150 000 öS, in der Computeranimation mit 300 000 öS dotiert ist; außerdem sind für jeden Bereich zwei Anerkennungspreise vorgesehen.

Am Wettbewerb teilnehmen können neben Einzelpersonen auch Gruppen, wissenschaftliche bzw. künstlerische Institute sowie Firmen. Interessierte senden ihre Beiträge bis spätestens 31. Mai 1987 an folgende Adresse:

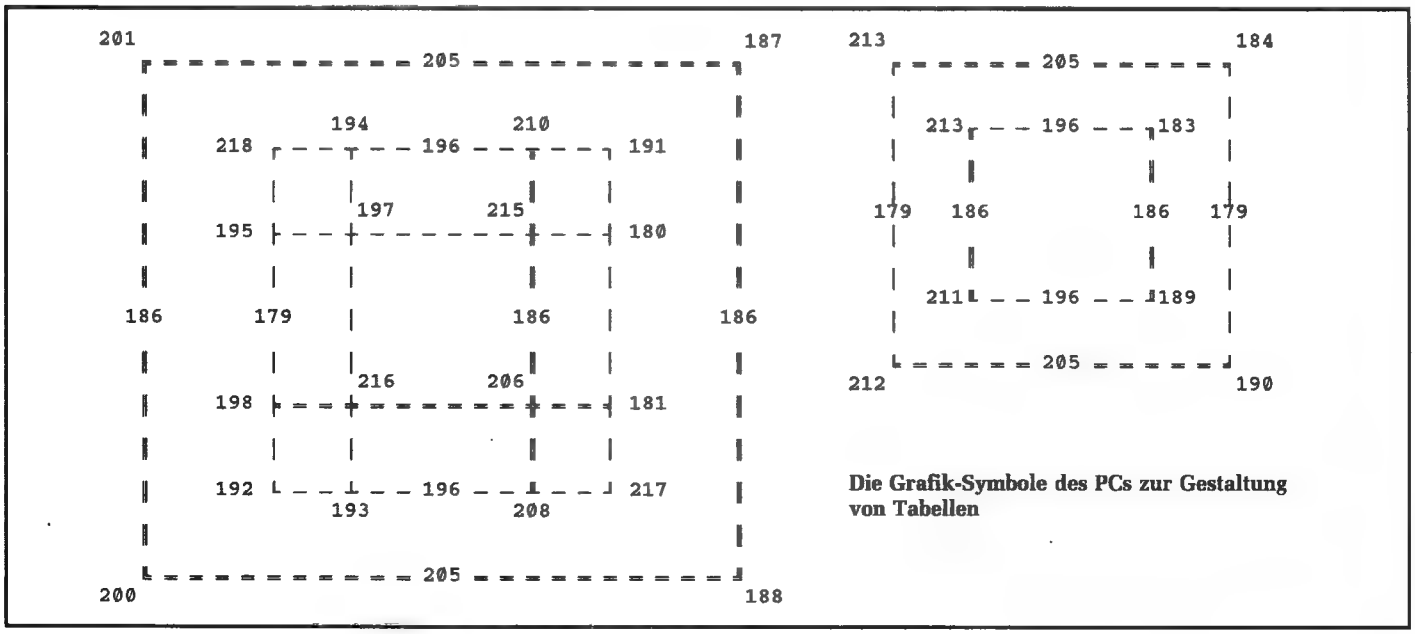
Prix Ars Electronica
ORF
Franckstr.2a
A-4010 Linz
Austria

Dort sind auch detaillierte Informationen über die Teilnahme-Modalitäten zu beziehen. mk

PC-Grafik-Symbole

Beim PC sind ab 80H (dez. 128) in der ASCII-Tabelle Grafik-Symbole enthalten, die zum Zeichnen von Tabellen im Textmodus recht brauchbar sind. Die Grafik-Symbole lassen sich durch Betätigen der Alt-Taste und der entsprechenden Dezimalzahl erzeugen (Zehner-Tastatur bzw. Cursor-Tastenblock!). Zum besseren Erkennen der einzelnen Symbole wurden im *Bild* Leerzeichen eingefügt.

Karl-Heinz Philippi



Die Grafik-Symbole des PCs zur Gestaltung von Tabellen

Rainer Walther

VIA 6522 am NDR-68008

Vor dem Anschluß des preiswerten und universellen Bausteins 6522 an ein 68008-System steht der Bau einer kleinen Schaltung zur Synchronisierung mit dem Systemtakt.

Der Anschluß des 6522 an den 68008 ist problematisch, da die VIA die Daten synchron zum Φ2-Takt, der 68008 dagegen asynchron überträgt. Zwar läßt sich der 68008 auf synchrone I/O-Bausteine umschalten, was jedoch auf der CPU-68K-Platine nicht berücksichtigt wurde. Somit benötigt man eine Zusatzschaltung, die den Zugriff auf die VIA mit dem Φ2-Takt synchronisiert (*Bild 1*).

Das Φ2-Signal (2 MHz) wird aus dem 8-MHz-Takt mit zwei D-Registern im 74LS175 erzeugt. Mit $\overline{\text{RESET}}$ wird über die Inverter I3 und I4 das Flipflop FF1 und somit auch FF2 bis FF4 gelöscht.

Bei einem I/O-Zugriff wird FF1 über den Vergleicher 74LS85 gesetzt. Dabei wird über I1 die $\overline{\text{WAIT}}$-Leitung am Bus aktiviert und der Prozessor angehalten.

Ebenso gibt der nun freigegebene Demultiplexer 74LS138 in Abhängigkeit der $\overline{\text{RD}}$- und $\overline{\text{WR}}$-Leitungen den Treiber 74LS245 beim Schreiben oder den Zwischenspeicher 74LS373 beim Lesen von Daten frei. Die Trennung der Treiber ist notwendig, da die Daten der VIA beim Lesen nur etwa 10 ns nach der fallenden Flanke von Φ2 stabil bleiben. Sie werden durch Φ2 gesteuert und für 500 ns im 74LS373 zwischengespeichert. Der Prozessor hat somit genügend Zeit zur Übernahme.

Das nun freigegebene FF2 wartet auf die fallende Flanke von Φ2 und gibt FF3 frei, ebenso über cs1 und $\overline{\text{cs2}}$ beide VIAs. Die Auswahl VIA-0 oder VIA-1 erfolgt über die Adreßleitung A4.

Nun beginnt der eigentliche Datentransfer. Die Adreß- und Steuerleitungen an der selektierten VIA bleiben für 500 ns stabil. Durch die steigende Flanke von Φ2 wird FF3 gesetzt und FF4 freigegeben. Bei der nächsten fallenden Flanke werden die Daten übertragen und FF4 gesetzt. Dabei wird über I2 FF1 gelöscht, die $\overline{\text{WAIT}}$-Leitung wird inaktiv, und nun kann auch der Prozessor den Datentransfer beenden.

Der VIA-Zugriff dauert folglich ein bis zwei Φ2-Takte, je nachdem, wann der

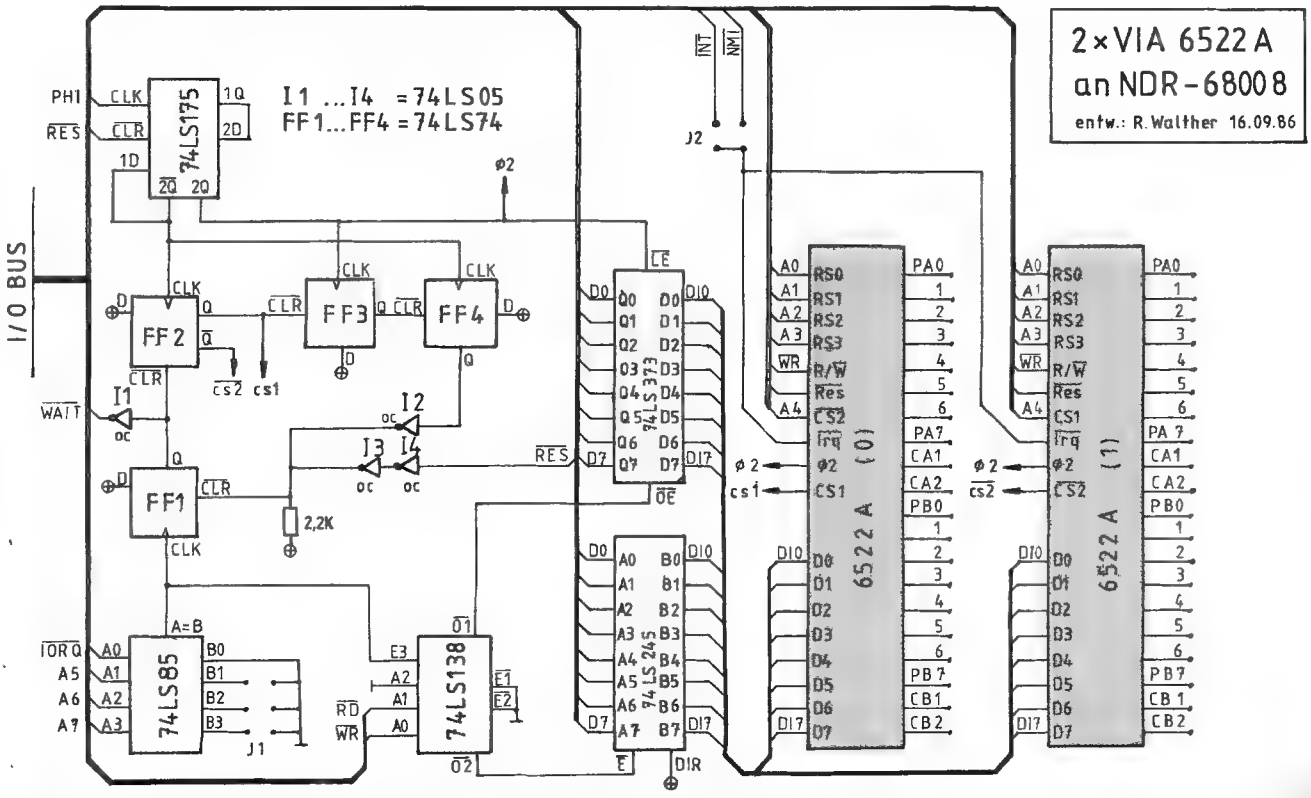


Bild 1. Diese Schaltung erlaubt den Betrieb zweier 6522 am NDR-68008

Prozessor in bezug auf Φ2 auf die Karte zugreift (siehe Timing-Diagramm, *Bild 2*). Man sollte deshalb nur die 2-MHz-VIAs 6522A verwenden, um die Zugriffszeit in Grenzen zu halten. Der Frequenzteiler für das Φ2-Signal ist ebenfalls für 2-MHz-VIAs ausgelegt. Will man trotzdem 1-MHz-Typen verwenden, müssen zwei weitere D-Register-Stufen eingefügt werden.
Die Adresse der Platine läßt sich mit J1 in 32-Byte-Schritten einstellen und sie belegt 32 Byte im I/O-Bereich des Prozessors.
Mit J2 kann man die Interrupt-Priorität einstellen. Will man keinen Interrupt erzeugen, so läßt man J2 offen.

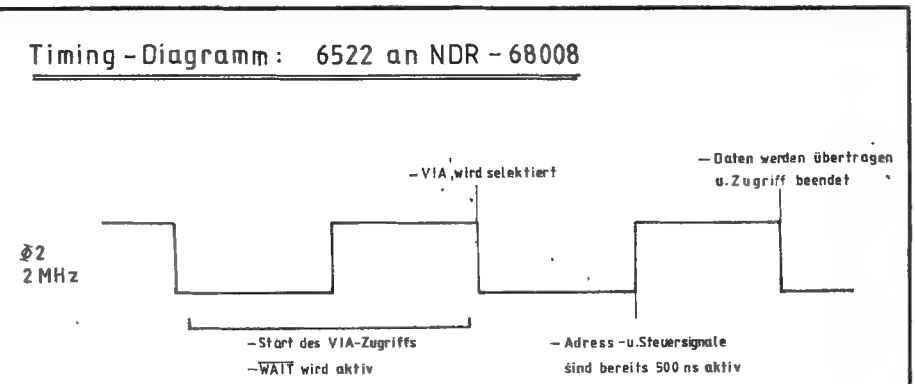


Bild 2. Zeitdiagramm der Schaltung

Der mc-68000-Computer lernt griechisch

Viele wissenschaftliche Formeln verwenden griechische Buchstaben. Um sie auf dem Bildschirm darstellen zu können, sind kleine Änderungen im Zeichensatz nötig. Das *Bild* zeigt eine Anpassung für den mc-68000-Computer im 80-Zeichen-Modus: Auf der linken Seite stehen die geänderten Inhalte des Zeichen-Generator-RAMs, rechts die neue Tastaturbelegung.

Falls mit beiden Zeichensätzen gearbeitet werden soll, so muß der aktuelle Zeichensatz von $FFF008...$FFF1D7 in einen freien RAM-Bereich gerettet und bei Bedarf zurückgeschrieben werden.

Alle im Bild nicht aufgeführten Zeichen (Zahlen und Sonderzeichen) wurden nicht geändert.

Georg Hausladen

$FFF008	$1818	$243C	$4266	$0000	A	→	A	$FFF108	$0000	$3A44	$443A	$0000	a	→	α
$FFF010	$7C22	$3C22	$227C	$0000	B	→	B	$FFF110	$3C42	$7C42	$427C	$4040	b	→	β
$FFF018	$1824	$5A5A	$2418	$0000	C	→	Θ	$FFF118	$1C24	$1EC4	$4438	$0000	c	→	θ
$FFF020	$1818	$2424	$427E	$0000	D	→	Δ	$FFF120	$1E20	$3C42	$423C	$0000	d	→	δ
$FFF028	$7E22	$3820	$227E	$0000	E	→	E	$FFF128	$0000	$7E3C	$403E	$0000	e	→	ε
$FFF030	$082C	$4A4A	$2C08	$0000	F	→	Φ	$FFF130	$0000	$344A	$4A3C	$0808	f	→	φ
$FFF038	$7E22	$2020	$2070	$0000	G	→	Γ	$FFF138	$0000	$4624	$1818	$2818	g	→	γ
$FFF040	$6642	$3C42	$4266	$0000	H	→	H	$FFF140	$0000	$5C62	$4242	$0202	h	→	η
$FFF048	$1C08	$0808	$081C	$0000	I	→	I	$FFF148	$0000	$4040	$423C	$0000	i	→	ι
$FFF050	$7E42	$3C00	$427E	$0000	J	→	Ξ	$FFF150	$3E40	$3C40	$403C	$027C	j	→	ξ
$FFF058	$7628	$3030	$2876	$0000	K	→	K	$FFF158	$0000	$6638	$3866	$0000	k	→	κ
$FFF060	$1818	$2424	$4266	$0000	L	→	Λ	$FFF160	$2050	$1010	$2846	$0000	l	→	λ
$FFF068	$4266	$5A5A	$4266	$0000	M	→	M	$FFF168	$0000	$4242	$427C	$4040	m	→	μ
$FFF070	$4662	$524A	$4662	$0000	N	→	N	$FFF170	$0000	$4644	$5860	$0000	n	→	ν
$FFF078	$1824	$4242	$2418	$0000	O	→	O	$FFF178	$0000	$3C42	$423C	$0000	o	→	ο
$FFF080	$7E24	$2424	$2466	$0000	P	→	Π	$FFF180	$0002	$3C64	$2466	$0000	p	→	π
$FFF088	$0000	$0000	$0000	$0000	Q	→	Blank	$FFF188	$0000	$0000	$0000	$0000	q	→	Blank
$FFF090	$7C22	$3C20	$2070	$0000	R	→	P	$FFF190	$0000	$3C42	$427C	$4040	r	→	ρ
$FFF098	$7E42	$1020	$227E	$0000	S	→	Σ	$FFF198	$0002	$3C44	$4438	$0000	s	→	σ
$FFF0A0	$3E2A	$0808	$081C	$0000	T	→	T	$FFF1A0	$0002	$3C48	$0A04	$0000	t	→	τ
$FFF0A8	$084A	$4A3C	$081C	$0000	U	→	Υ	$FFF1A8	$0008	$6A2A	$2A1C	$0808	u	→	ψ
$FFF0B0	$3C42	$2418	$5A7E	$0000	V	→	Ω	$FFF1B0	$0000	$2A4A	$4A34	$0000	v	→	ω
$FFF0B8	$0000	$0000	$0000	$0000	W	→	Blank	$FFF1B8	$0000	$0000	$0000	$0000	w	→	Blank
$FFF0C0	$6624	$1818	$2466	$0000	X	→	X	$FFF1C0	$0000	$6224	$1818	$2426	x	→	χ
$FFF0C8	$7622	$1408	$081C	$0000	Y	→	Y	$FFF1C8	$0000	$6622	$221C	$0000	y	→	υ
$FFF0D0	$7E44	$0810	$227E	$0000	Z	→	Z	$FFF1D0	$7E10	$2040	$403C	$027C	z	→	ζ

Mit diesen Speicherinhalten bringt der mc-68000-Computer griechische Buchstaben auf den Bildschirm

H. Nikolaus Schaller

Z80-Karte für den mc-68000-Computer

Teil 1: Die Hardware

Für so manchen Besitzer des mc-68000-Computers unter CP/M-68K ist der Rückgriff auf die große CP/M-80-Programmbibliothek durchaus noch interessant, manch anderer hätte gerne einen unabhängigen Subrechner. Daher wird in diesem Beitrag eine Einsteck-Karte mit dem altbewährten (und preiswerten) 8-MHz-Z80 vorgestellt, auf der CP/M-80-Software lauffähig ist und die als unabhängiger Peripheriерechner eingesetzt werden kann.

Mit der Schaltung wurden also zwei Ziele verfolgt: Zum einen soll die Platine eine schnelle Verarbeitung von CP/M-80-Programmen ermöglichen, zum anderen soll sie als Peripherie-Controller verwendet werden können. Daher wurden einige Schaltungsdetails nötig, die nur bei Verwendung der Peripheriebausteine eingebaut werden müssen. Außerdem ist ein Lochrasterfeld mit 400 Punkten vorhanden, das vom Anwender frei verdrahtet werden kann. Die Platine ist also in der Lage, Prozeßsteuerungen als autonomer Kontrollrechner zu übernehmen, wobei ihr Programm vom mc-68000-Computer gestartet wird. Als Entwicklungsumgebung sind auch hier die CP/M-80-Programme sehr nützlich.
Im ersten Teil wird die Hardware beschrieben, wobei nur die Besonderheiten betont werden. Ansonsten wird auf die Fülle der im Fachbuchhandel erhältlichen Z80-Hardwarebücher verwiesen. Im zweiten Teil geht es dann darum, wie die Platine aufzubauen ist und wie man sie testet. Auch die Prinzipien der Software, die benötigt wird, um die Platine zu betreiben, werden im nächsten Heft geschildert, ebenso wie Funktion und Bedienung der Programme für die CP/M-80-Emulation.

Takt und Reset

Für die Takterzeugung kommt ein 8-MHz-, 12-MHz- oder 16-MHz-Quarz-Oszillatormodul zum Einsatz, um hohe Schwingsicherheit zu erreichen (*Bild 1*). Sein Ausgangssignal wird durch ein Flipflop halbiert, was einen sauberen, symmetrischen Takt sichert. Ein etwa 15 µs dauernder Resetimpuls kann über den Bus oder über Wortzugriffe ausgelöst werden. Der Impuls aktiviert das EPROM und sperrt $\overline{IRQ2}$.
Ein Flipflop (IC 24) schaltet das BOOT-EPROM (*Bild 2*) ein und aus; bei aktivem EPROM wird die untere RAM-Hälfte ausgeblendet. Die etwas ungewöhnliche Beschaltung der $\overline{CE}$, $\overline{PGM}$ und $\overline{OE}$-Anschlüsse ist gewollt: Ein Z80H-Prozessor erlaubt nämlich während eines $\overline{M1}$-Zyklus nur etwa 150 ns Zugriffszeit zwischen $\overline{MREQ}$ und stabilen Daten, zwischen Adressen und Datenzugriff bleiben dagegen etwa 200 ns. Die Zugriffszeit eines EPROMs wird immer von dem Zeitpunkt gemessen, ab dem die Adressen bei aktivem $\overline{CE}$-Signal stabil sind. Bei einem 2764-20 sind das eben 200 ns. Die Signale $\overline{OE}$ und (leider nicht bei allen Herstellern) $\overline{PGM}$ steuern die Ausgangstreiber direkt, und haben daher ei-

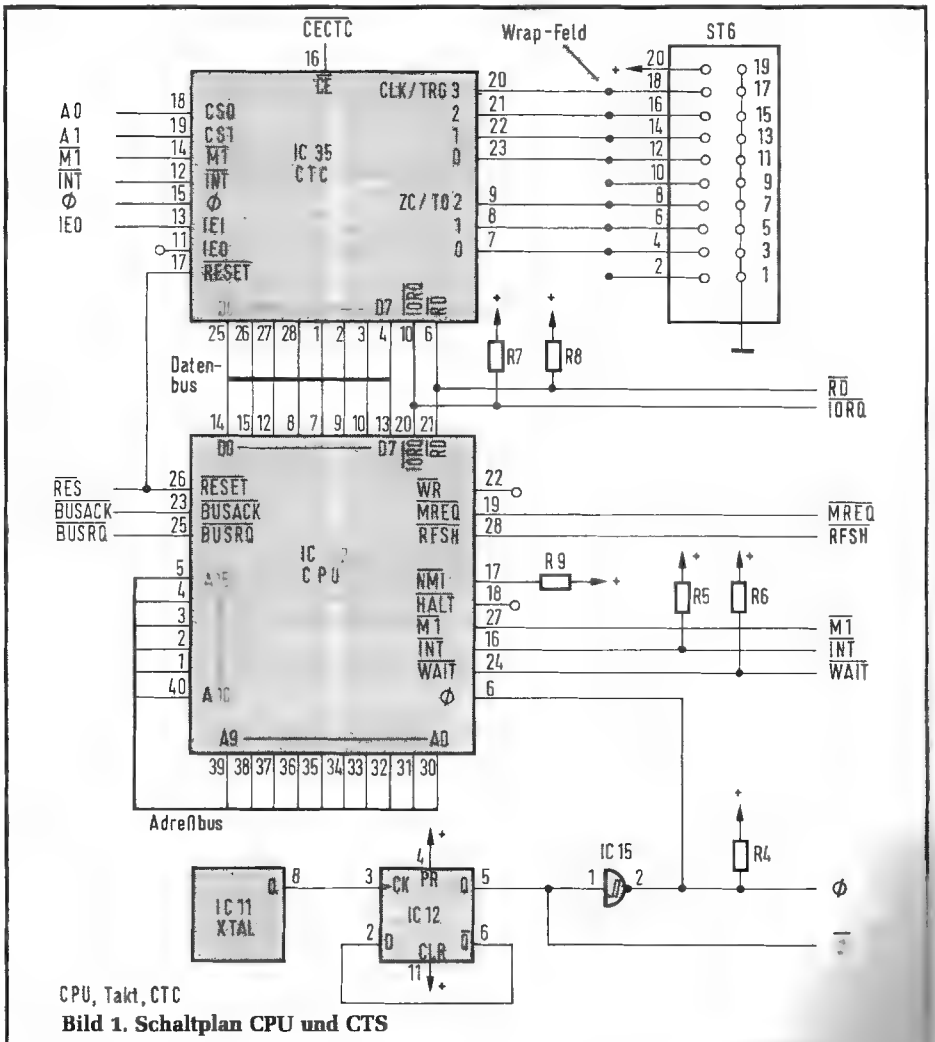


Bild 1. Schaltplan CPU und CTS

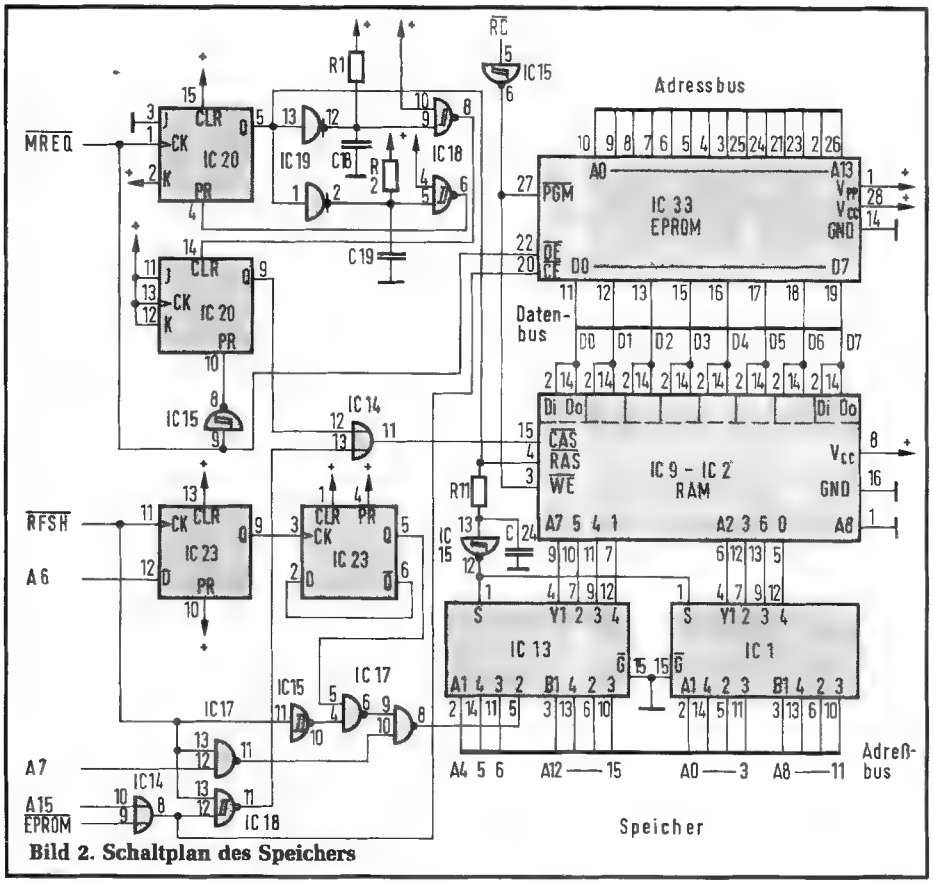


Bild 2. Schaltplan des Speichers

Tabelle 1: Bauteile für verschiedene Taktfrequenzen
Höherwertige Bauteile können natürlich eingesetzt werden.

Bauteil	CPU-Taktfrequenz: 4MHz	6MHz	8MHz
IC 2...9	4164-15	4164-15	4164-12
IC 11	8MHz-XTAL	12MHz-XTAL	16MHz-XTAL
IC 32	Z80A-CPU	Z80B-CPU	Z80H-CPU
IC 33	2764-25	2764-25	2764-20
IC 35	Z80A-CTC	Z80B-CTC	nicht verfügbar
IC 36	Z80A-SIO	Z80B-SIO	nicht verfügbar
IC 37	Z80A-PIO	Z80B-PIO	nicht verfügbar
C 24	100 pF	100 pF	100 pF
R 1	374 Ω	374 Ω	340 Ω
R 2	432 Ω	432 Ω	357 Ω
R 11	100 Ω	100 Ω	100 Ω

R1 und R2 hängen nur von der RAM-Zugriffszeit ab, nicht von der CPU-Taktfrequenz.

Die folgenden, durch diese Schaltung gegebenen Grenzwerte können die vom Hersteller gegebenen Grenzen weiter einschränken:

CPU-Takt: / RAM-Zugriff:	4 MHz 150	6 MHz 150	8 MHz 120	
RASmin	150	150	120	RAS=low
RPmin	100	100	90	RAS=high
RCDmin	30	30	30	RAS=low...CAS=low
RSHmin	75	75	60	RAS und CAS=low
RAHmin	20	20	20	Adressenwechsel nach RAS high-low
RAHmax nicht nach CAS=low				

Alle Zeitangaben in ns.

ne wesentlich geringere Verzögerung zur Folge (hier etwa 80 ns). Werden die langen EPROM-Zeiten mit den langen CPU-Zeiten (Adressen, $\overline{CE}$) und die kurzen EPROM- und CPU-Zeiten ($\overline{MREQ}$, $\overline{RD}$, $\overline{PGM}$, $\overline{OE}$) miteinander kombiniert, ist ein sicherer Betrieb gewährleistet. Ähnliche Überlegungen zeigen, daß für einen Z80B eine EPROM-Zugriffszeit von 250 ns ausreicht.

Viel Peripherie

Als Peripherie sind vier Timer (CTC), zwei 8-bit-Parallel-Schnittstellen (PIO) und zwei asynchrone oder synchrone serielle Kanäle (SIO) vorhanden (*Bild 3*). Durch IC 26 werden A6 und A7 decodiert, um entweder einen der drei Peripheriebausteine oder das Kontrollregister auszuwählen. Bei Adressierung dieses Registers werden A0 und A1 in zwei Flipflops übernommen. Dabei schaltet A0 das EPROM ein oder aus. A1 fordert nach Synchronisierung mit dem E-Takt des 68000 einen Interrupt auf Level 2 der 68000-CPU an oder nimmt ihn zurück.
Alle drei Peripheriebausteine können über die Interruptkette einen Interrupt (Mode 2) bei der Z80-CPU anfordern. R12 und die Dioden D1, D2 bilden ein UND-Gatter, das verhindert, daß während der Interruptbestätigung die Flipflops in IC24 unbeabsichtigt schalten, wenn der Programmzähler auf einer Adresse im Bereich 0XXC0H bis 0XXFFH steht. Die Steckerbelegung aller Schnittstellen ist in *Tabelle 4* angegeben. Dabei gilt folgende Regel: die ungeradzahligen Pins liegen alle auf Masse, um das Übersprechen bei Flachbandkabeln zu reduzieren, die geradzahligen führen die Signale.

Der dynamische Speicher

Jede fallende Flanke am $\overline{MREQ}$-Anschluß des Z80 signalisiert den Anfang eines Speicherzugriffs (Memory Request). Dadurch wird das $\overline{RAS}$-Flipflop in IC 20 gesetzt. Da hierfür ein Schottky-TTL-Baustein verwendet wird, ist die Durchlaufverzögerung sehr klein (typisch 5 ns). Nun kann man aber nicht einfach warten, bis die CPU das $\overline{MREQ}$-Signal zurücknimmt, denn dann würde die Zeit bis zum nächsten $\overline{MREQ}$ im $\overline{M1}$-Zyklus (ein Lesezugriff und direkt anschließend ein Refreshzugriff) nicht ausreichen, um die Daten im RAM zurückzuschreiben („Precharge"-Zeit). Daher muß man $\overline{RAS}$ vorzeitig beenden.

IC 19, R 2, C 19 und IC 18 erfüllen diese Funktion. Wenn $\overline{RAS}$ aktiv wird, gibt der Ausgangstransistor in IC 19 den Kondensator frei, der sich jetzt über R 2 auflädt. Wenn die Spannung die obere Schwelle des Schmitt-Triggers erreicht, setzt der das $\overline{RAS}$-Flipflop zurück. Dadurch wird $\overline{RAS}$ vor $\overline{MREQ}$ inaktiv. Außerdem wird der Kondensator über den niedrigen Ausgangswiderstand des IC 19 sehr schnell entladen. Die Zeit, die $\overline{RAS}$ aktiv bleibt (t_{RAS}) darf aber nicht zu kurz werden. Sie muß daher an die gewählten Speicherbausteine angepaßt werden, doch dazu später.
Die Funktion dieser Schaltung ist also ein Monoflop mit extrem geringer Durchlaufverzögerung und kurzer Erholungszeit, damit die Länge des nächsten Ausgangsimpulses nicht zu sehr von der Vorgeschichte abhängt. Mit dieser Schaltung können alle Forderungen erfüllt werden, die ein $\overline{RAS}$-Signal und ein Z80 mit 8 MHz Takt für zuverlässigen Betrieb stellen.

Für $\overline{CAS}$ wird eine ähnliche Schaltung verwendet. Hier bestimmen R1 und C18 die Zeit (t_{RCD}) von $\overline{RAS}$, bis $\overline{CAS}$ aktiv wird. Dieses Signal wird erst wieder inaktiv, wenn der aktuelle Zugriff beendet ist. An den Speicher wird $\overline{CAS}$ allerdings nur dann weitergegeben, wenn kein Refreshzyklus oder EPROM-Zugriff erfolgt. Diese Zugriffe werden dadurch als $\overline{RAS}$-only-Zyklen durchgeführt. Das Zeitdiagramm in Bild 5 stellt diese Beziehungen noch einmal dar.

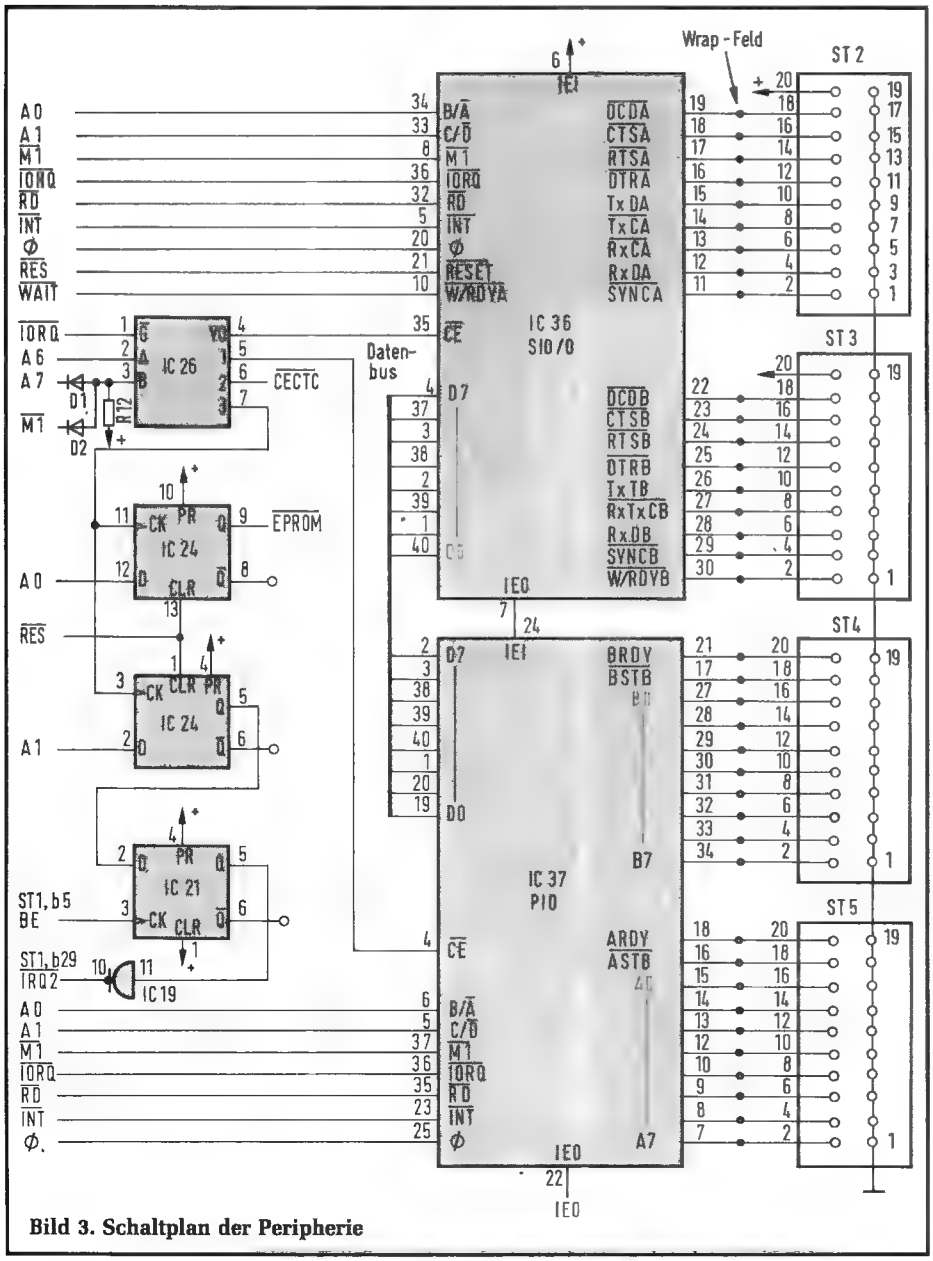


Bild 3. Schaltplan der Peripherie

Die Adressen werden durch IC 1 und IC 13 gemultiplext und an die Speicherbausteine weitergegeben. Die Umschaltung von Reihen auf Spaltenadressen wird von $\overline{RAS}$ nach einer kleinen Verzögerung durch R 11 und C 24 (t_{RAH}) gesteuert.
Der Prozessor Z80 besitzt einen eingebauten Refreshzähler, der aber nur 7 Adreßbits zur Verfügung stellt. Um auch Speicher verwenden zu können, die 256 Reihen beim Refresh brauchen, muß das achte Bit in einer externen Schaltung erzeugt werden. Dazu wird während jedes Refreshzyklus, erkennbar am aktiven $\overline{RFSH}$, der Zustand von A6 zwischengespeichert. Mit dem Ausgang dieses Flipflops kann man nun ein zweites triggern, wie bei einem Asynchronzähler, und hat dadurch den Zähler erweitert. IC 23 übernimmt hier diese Funktion. Durch IC 17 wird das vom Z80 während der Refreshzeit ausgegebene Signal durch dieses erweiterte Refreshbit ersetzt.
Weil das invertierte $\overline{RD}$-Signal dieser Schaltung als Schreibsignal („write enable") für die Speicher verwendet wird, kommen nur Schreibzyklen vor, bei denen $\overline{WE}$ vor $\overline{CAS}$ aktiv ist („early write"), daher dürfen Datenein- und Datenausgänge zusammengeschaltet werden. Sie hängen deshalb auch direkt am internen Z80-Bus.

Dual-Port-Memory

Zunächst einmal das grobe Prinzip: Zugriffe des mc-68000-Computers auf den Z80-Speicher werden per DMA durchgeführt. Dazu muß der Z80 als erstes seinen internen Bus freigeben. Dann kann die Schaltung die Adressen- und Datentreiber aktivieren, um den 68000-Bus mit den Z80-Speichern zu verbinden. Als nächstes kann ein Zugriff auf die Speicher gestartet werden. Wenn dieses geschehen ist, muß dem 68000 mitgeteilt werden, daß er den Zyklus beenden kann. Erst wenn der Zyklus beendet ist, kann der Z80 weiterarbeiten.
Wenn der 68000 einen Speicherzugriff durchführen will, teilt er das über das $\overline{SSEL}$-Signal mit (*Bild 4*). Dieses Signal führt über IC 14 an die Busanforderungsleitung ($\overline{BUSRQ}$) des Z80. Sobald der Z80 seinen laufenden Zugriff beendet hat, macht er die Adressen und Daten hochohmig. Diesen Zustand teilt er über $\overline{BUSACK}$ mit. Um Refresh und kurze Interrupt-Antwortzeiten zu garantieren sowie fehlerhafte Zugriffe zu vermeiden, wird der Bus nach jedem 68000-Zugriff wieder an den Z80 übergeben. Dazu

Tabelle 2: Adressen beim Zugriff auf den Z80-Speicher

Zugriff	A0 (intern)	UDS	LDS
keiner	–	1	1
Byte, gerade	0	0	1
Byte, ungerade	1	1	0
Wort, gerade	0	0	0
Wort, ungerade	1	gibt	Adreß-fehler

wird ein Flipflop (IC 21) am Ende eines 68000-Zugriffs gesetzt. Es sperrt nun den nächsten Zugriff, bis der Z80 den Bus sicher freigegeben hat, also jetzt meistens einen eigenen Zyklus durchführt, an dessen Ende er erneut auf $\overline{BUSRQ}$ reagieren kann. Wegen dieser Schaltung kann ein 68010 im Loop-Mode den Z80 nicht dauerhaft anhalten, der RAM-Refresh kann daher nicht ausfallen.
Ab dem Zeitpunkt, an dem Adressen und Steuerleitungen hochohmig sind, vergeht einige Zeit, bis $\overline{BUSACK}$ aktiv wird. Damit die Speichersteuerung dadurch nicht durcheinanderkommt, ziehen Pullup-Widerstände $\overline{RD}$, $\overline{MREQ}$ und $\overline{IORQ}$ in den inaktiven Zustand.
Wenn der Z80 $\overline{BUSACK}$ aktiviert hat, werden $\overline{RD}$ und $\overline{MREQ}$ zugeschaltet. Gleichzeitig werden die Adressentreiber eingeschaltet, die die 68000-Zugriffsadresse auf den Z80-Bus geben. Adresse A0, die es beim 68000 nicht gibt, wird über einen kleinen Trick erzeugt (*Tabelle 2*).
Wie man sieht, entspricht $\overline{UDS}$ der internen Adresse A0, unabhängig vom Zugriff. Also kann man $\overline{UDS}$ direkt als A0 verwenden, wenn man beachtet, daß die Datenstrobes wesentlich später aktiv werden als die Adressen, bei Schreibzugriffen sogar später als $\overline{AS}$ selbst. Bei dieser Schaltung ist durch die Gatterlaufzeiten und die – noch zu beschreibende – Synchronisation gesichert, daß auch A0 einige Zeit vor $\overline{MREQ}$ stabil wird, was für t_{ASR} (address setup time for rows) wichtig ist.
Das Z80-Signal $\overline{WR}$ braucht nicht erzeugt werden, da es in der gesamten Schaltung nicht verwendet wird. $\overline{RD}$ entsteht aus dem 68000-Signal BR/$\overline{W}$. Es muß dazu einmal invertiert werden, was auch einen günstigen Nebeneffekt hat: jede 68000-Busleitung sollte mit nur einem LS-TTL-Eingang belastet werden. Da auch beide Datentreiber über BR/$\overline{W}$ in der Richtung umgeschaltet werden müssen, dient der Inverter nebenbei als Treiber.
Aus dem gleichen Grund werden auch $\overline{BUDS}$ und $\overline{BLDS}$ über zwei Treiber geführt. Sie steuern dann einen Dekoder (IC 26), der feststellt, ob der 68000 keinen Zugriff, einen Byte- oder gar einen Wortzugriff anfordert. Bei einem Wortzugriff wird ein Busfehler ausgelöst, da die Z80-Karte solche Zugriffe nicht ausführen kann. Diese Situation wird gleichzeitig benutzt, um für CPU und Peripherie einen Reset auszulösen. Damit kann man den Z80, durch 68000-Software gesteuert, in einen definierten Anfangszustand bringen.
Bei einem Bytezugriff gibt der Dekoder Auskunft, auf welcher Hälfte der 68000-Datenleitungen übertragen werden soll. Dadurch wird der passende Datentreiber freigegeben.
Wurde einer der beiden Treiber eingeschaltet, also ein gültiger Zugriff begonnen, werden die Flipflops im IC 22 freigegeben. Bei der nächsten steigenden Flanke des Z80-Takts wird ein $\overline{MREQ}$ erzeugt, der den EPROM- oder RAM-Zugriff startet. Das ist der gleiche Zeitpunkt, zu dem der Z80 seine eigenen Zugriffe beginnt. Einen Taktimpuls später, je nach Taktfrequenz also nach 125...166 ns, wird $\overline{DTACK}$ aktiv, das

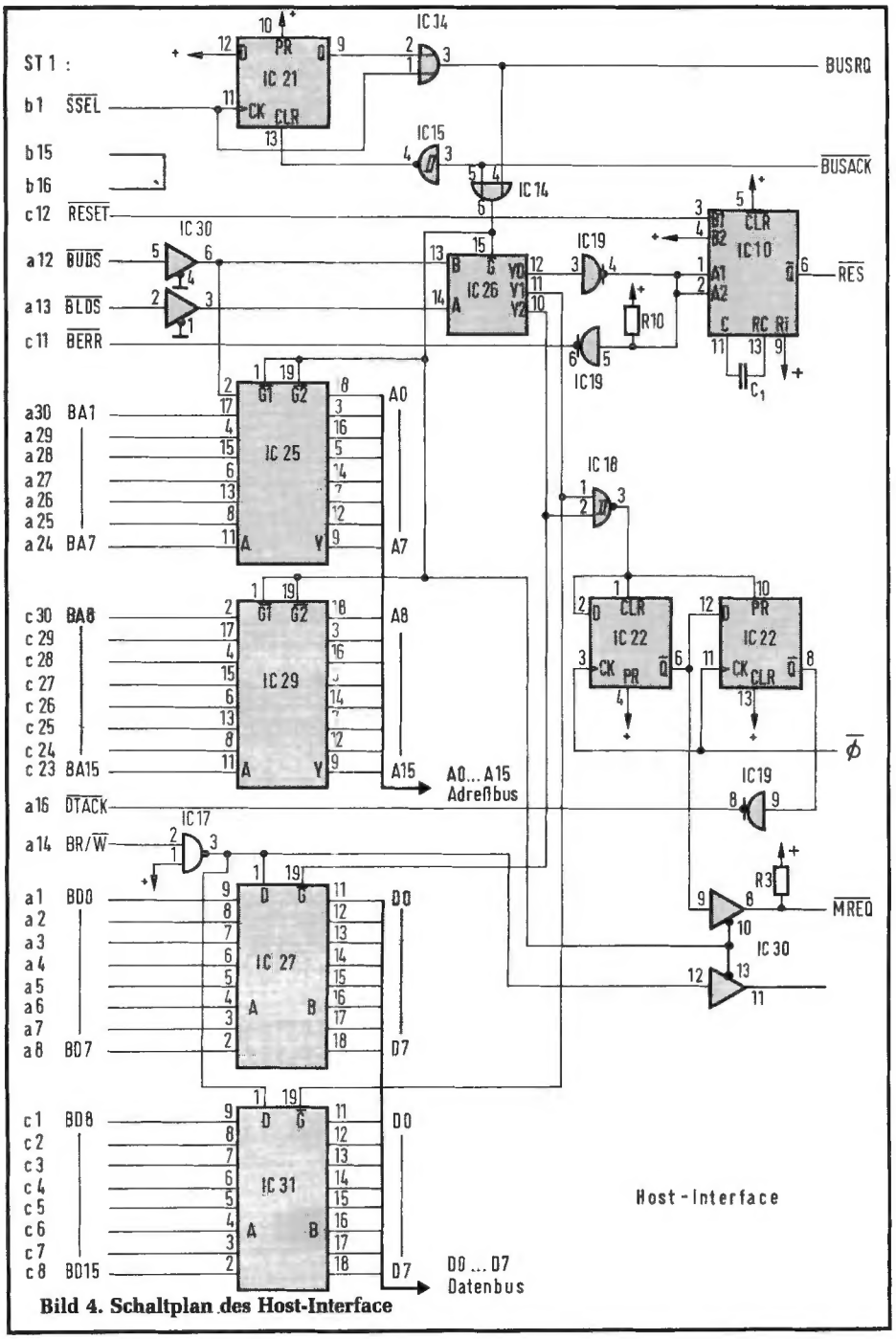
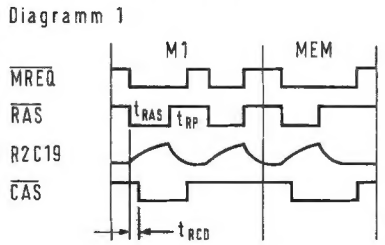


Bild 4. Schaltplan des Host-Interface



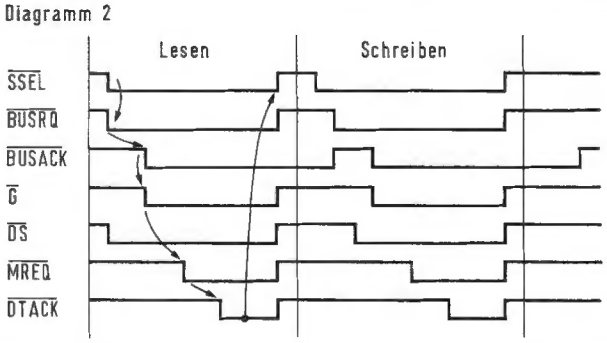


Bild 5. Zeitdiagramme des Speicherzugriffs und des DMA-Ablaufs

dem 68000 mitteilt, daß der Zugriff läuft. Diese Verzögerung dauert alleine schon fast länger, als der Z80 für einen ganzen M1-Zyklus braucht. Außerdem benötigt der 68000 einige Zeit, bis er auf DTACK reagiert.
Am Ende eines Zugriffs durch den 68000 werden SSEL und der Datenstrobe inaktiv, dadurch wird das MREQ-Flipflop sofort gelöscht, womit der Zugriff auf den Speicher beendet ist. Daß dieses asynchron geschieht stört nicht, da keiner der Speicher auf den CPU-Takt angewiesen ist. Die taktsynchrone Z80-Peripherie kann aus Gründen der Systemzuverlässigkeit und Schaltungsvereinfachungen sowieso nicht über DMA angesprochen werden.
Mit SSEL wird auch BUSRQ und kurz darauf BUSACK inaktiv. Als Folge davon werden die Daten und Adressentreiber wieder abgeschaltet. Jetzt kann der ganze Ablauf von vorne beginnen. *Bild 5* zeigt noch einmal den zeitlichen Ablauf eines DMA-Zugriffs.
Diese Steuerung kann auch die 68000-Zyklen beim TAS-Befehl („test and set") ausführen. In diesem Fall bleibt der Z80 angehalten, weil SSEL während des ganzen Zyklus aktiv bleibt. Für beide Datenzyklen werden aus den BUDS- oder BLDS-Impulsen jeweils zwei Anforderungen (MREQ) für den Speicher und zwei DTACK-Impulse für den 68000 erzeugt.

Tabelle 3: Belegung des Wrap-Feldes
Blick auf Platinenlötseite, Busstecker unten rechts

obere Zeile von links: PIO	rechte Spalte von oben nach unten: (SIO/0)	untere Zeile von rechts: CTC
GND	BRDY	(ST6, Pin2)
A7	WRDYB	ZCTO0
A6	SYNCB	ZCTO1
A5	RxDB	ZCTO2
A4	RxTxCB	(ST6, Pin10)
A3	TxDB	C/T0
A2	DTRB	C/T1
A1	RTSB	C/T2
A0	CTSB	C/T3
ASTB	DCDB	10× -NC-
ARDY	SYNCA	GND
B7	RxDA	
B6	RxCA	
B5	TxCA	
B4	TxDA	
B3	DTRA	
B2	RTSA	
B1	CTSA	
B0	DCDA	
BSTB	(ST6, Pin2)	
BRDY		

linke Spalte: GND

zweite Spalte von rechts, 6. Zeile von unten: VCC

Bauteile

Zunächst einige Hinweise zur Anwendung der Peripherie: bei SIO und CTC wurde die Betriebsspannung auf ST2, ST3 und ST6 geführt, man kann also kleinere Schaltungen über diesen Stekker versorgen. Allerdings ist hier Vorsicht geboten, da die Entfernung zum Netzteil ziemlich groß werden kann, und wegen kleinen Leiterquerschnitten große Spanungsabfälle und damit Störungen nicht auszuschließen sind.
Als Verbindungskabel kann man Flachbandkabel an die Buchsen quetschen. Die Kabellänge sollte wegen Leitungskapazitäten und -induktivitäten möglichst gering bleiben.
Auf dem Wrapfeld lassen sich mehrere 16polige ICs unterbringen. Werden dabei Wrap-Fassungen eingesetzt, so empfiehlt es sich, Stiftreihen an die Positionen auf der Platinenunterseite zu löten, an denen die Peripheriesignale angeschlossen sind. Daran kann man die Betriebsspannung und die Ports anschließen. Die Belegung ist *Tabelle 3* zu entnehmen. Denkbare Anwendungen dieses Feldes sind z.B. V.24-Treiber oder einige Reed-Relais.

Eine wichtige Frage ist, ob überhaupt Peripherie verwendet werden soll: Ohne Peripherie kann der Z80H mit vollen 8 MHz laufen. Mit Peripherie muß der CPU-Takt auf 6 MHz oder weniger festgelegt werden, da es bisher keine Z80H-Peripheriebausteine gibt.
Werden Schnittstellen eingesetzt, so ist jede Kombination aus SIO, PIO und CTC möglich. Beim SIO-Baustein gibt es drei verschiedene Pinbelegungsoptionen, doch ist hier eine spätere Änderung möglich, da sich nur die Belegungen des Kanals B unterscheiden. Auswirkungen hat das nur auf den Stecker ST3 (*Tabelle 4*) und die Anschlüsse am Wrapfeld.
Es müssen aber keineswegs alle drei Plätze bestückt sein, hier können sogar die Fassungen, ebenso die Stecker, fehlen. Und ob gerade oder abgewinkelte Stecker verwendet werden, bleibt jedem selbst überlassen. Bei fehlender SIO oder PIO muß aber eine Drahtbrücke eingebaut werden, um die Interruptkette nicht zu unterbrechen.
Beim SIO-Kanal A gibt es die Möglichkeit, den Prozessor während I/O-Zugriffen anzuhalten, bis wirklich Daten übertragen werden können. Dazu muß das SIO-Schreibregister 1 entsprechend programmiert werden. Allerdings bleibt mit der CPU auch der Refresh für die dynamischen Speicher stehen, man sollte diese Betriebsart nur dann einsetzen, wenn die Datenrate so hoch ist, daß ein Programm mit Polling oder Interrupt nicht mehr mitkommt. Das entsprechende Signal vom Kanal B wird auf den Peripheriestecker ST3 geführt, da es bei einigen Anwendungen für die Peripherie selbst nützlich sein kann.
Ist die CPU-Taktfrequenz festgelegt, benötigt man ein Quarzoszillatormodul, das auf der doppelten Frequenz schwingt. Abhängig davon bestimmen sich die Zugriffszeiten der Speicher. Über 6 MHz Taktfrequenz benötigt das EPROM eine Zugriffszeit von 200 ns. Bei niedrigerer Taktfrequenz reichen 250 ns völlig aus. Eine wichtige Einschränkung

der Typen ergibt sich daraus, daß das $\overline{PGM}$-Signal unbedingt parallel zu $\overline{OE}$ verarbeitet werden muß, und nicht wie $\overline{CE}$, also eine kurze Zugriffszeit ergibt. Meistens schweigt sich das Datenblatt darüber aus. Folgende Typen sind nach Messungen oder laut Datenblatt geeignet: D2764-2 (NEC), HN482764-2 (Hitachi), MBM2764-20 (Fujitsu), TMM2764-2 und TMM27128-2 (Toshiba). Weitere Typen können ebenfalls geeignet sein, Auskunft darüber gibt vielleicht doch das Datenblatt oder – in jedem Fall – eine Messung.
Für größere, eigene Programme ist vorgesehen, daß ein 27128 eingesetzt wird, der dann 16 KByte statt acht zur Verfügung stellt. Hier gelten die gleichen Anforderungen an die Zugriffszeit. Drahtbrücken oder Jumper sind dazu nicht zu ändern, da beide EPROM-Typen eine identische Anschlußbelegung besitzen. Ebenfalls von der Taktfrequenz der CPU hängt die benötigte Zugriffszeit der dynamischen Speicher ab. Sie muß bei 8 MHz 120 ns betragen, bei 6 MHz reichen 150 ns aus. Die Typenbezeichnungen können von Hersteller zu Hersteller stark abweichen. Folgende Liste gibt eine Auswahl wieder: MB8264A, HM4864A, 2164A (Intel), TMS4164, uPD4164 bzw D4164 (NEC). Die Bausteine können bei dieser Schaltung 128 oder 256 Refreshreihen haben. Wenn sie die Möglichkeit des „Hidden refresh" bieten, schadet das nicht, diese Funktion wird nur nicht ausgenutzt. Allerdings darf kein Speicher mit Autorefresh (d. h. mit eingebautem Zähler) verwendet werden, da die Steuerlogik diesen nicht unterstützt. Die Platine läßt sich auch mit 256-Kbit-Chips betreiben, wovon aber nur 64 Kbit genutzt werden.

Von der Speicherzugriffszeit hängt wiederum die Dimensionierung der Monoflops ab. Die Werte für R1, R2, R11 und C24 entnehmen Sie bitte Tabelle 1. Für die Kondensatoren kommen ausschließlich gute Metallfolienkondensatoren (Styroflex/MKS) in Frage, da nur sie eine geringe Toleranz und einen kleinen Temperaturkoeffizienten aufweisen. Ähnliches gilt für die Widerstände. Sie sollten unbedingt eine Genauigkeit von 1 % haben. Falls Sie sich nicht daran halten, ist nicht sichergestellt, daß die Schaltung über den gesamten Temperaturbereich einwandfrei funktioniert. Die übrigen Widerstände und Kondensatoren sind unkritisch, Toleranzen von 10 % stören nicht.

Für IC 20 muß unbedingt ein Schottky-TTL-Baustein verwendet werden, da nur diese Bausteine ausreichende Geschwindigkeit und Ausgangsleistung haben.
Bei schlechten IC-Fassungen können im Laufe der Zeit Wackelkontakte auftreten. Auch müssen die Fassungen anreihbar sein, da die Speicherbausteine dicht an dicht liegen. Eine Zusammenfassung der Möglichkeiten geben *Tabelle 5* (Bauteile), Tabelle 4 (Stecker) und Tabelle 1 (Alternativen).
Beim Aufbau sollte man sinnvollerweise die folgenden Reihenfolge einhalten: Man beginnt mit den flachesten Bauteilen – das dürften die Widerstände sein, eventuell auch die Kondensatoren. Bei den IC-Fassungen werden die größten zuerst eingesetzt, sie passen nur an die richtigen Positionen. Achten Sie darauf, daß die Fassungen sauber auf der Platine aufliegen.
Sind alle IC-Fassungen eingesetzt, lötet man die hohen Teile ein. Bei Widerständen und Kondensatoren muß vorher genau gepüft werden, ob die Werte stimmen (Farbringe, aufgedruckte Angaben oder eventuell mit einem Meßgerät): Bauteile wieder auszulöten, ist bei zweiseitigen Platinen etwas gefährlich, da die Durchkontaktierungen beschädigt werden können. Verpolte Tantalkonden-

Tabelle 4: Belegung der Peripherie-Stecker

Blick auf die (abgewinkelten) Stecker, Platine unten

ST2: Z80-SIO Kanal A

2	4	6	8	10	12	14	16	18	20
SYNCA	RxDA	RxCA	TxCA	TxDA	DTRA	RTSA	CTSA	DCDA	Vcc
GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
1	3	5	7	9	11	13	15	17	19

ST3: Z80-SIO Kanal B

2	4	6	8	10	12	14	16	18	20
SIO/0:									
WRDYB	SYNCB	RxDB	RxTxCB	TxDB	DTRB	RTSB	CTSB	DCDB	Vcc
SIO/1:									
WRDYB	SYNCB	RxDB	RxCB	TxCB	TxDB	RTSB	CTSB	DCDB	Vcc
SIO/2:									
WRDYB	RxDB	RxCB	TxCB	TxDB	DTRB	RTSB	CTSB	DCDB	Vcc
GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
1	3	5	7	9	11	13	15	17	19

ST4: Z80-PIO Kanal B

2	4	6	8	10	12	14	16	18	20
B7	B6	B5	B4	B3	B2	B1	B0	BSTB	BRDY
GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
1	3	5	7	9	11	13	15	17	19

ST5: Z80-PIO Kanal A

2	4	6	8	10	12	14	16	18	20
A7	A6	A5	A4	A3	A2	A1	A0	ASTB	ARDY
GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
1	3	5	7	9	11	13	15	17	19

ST6: Z80-CTC

2	4	6	8	10	12	14	16	18	20
–	ZCTO0	ZCTO1	ZCTO2	–	C/T0	C/T1	C/T2	C/T3	Vcc
GND	GND	GND	GND	GND	GND	GND	GND	GND	GND
1	3	5	7	9	11	13	15	17	19

satoren sind unbedingt zu vermeiden. D1, D2 und R12 fehlen im Platinenlayout und müssen daher nachträglich eingebaut werden, sofern Z80-Interrupts verwendet werden. Alle Bauteile finden auf der Platinenunterseite Platz. Auf der Lötseite ist zunächst IC26 Pin 3 von der Adreßleitung A7 abzutrennen (Leiterbahn führt direkt am IC vorbei). Über R12 wird Pin 3 mit Pin 16 (V_{cc}) verbunden. Die Anoden beider Dioden kommen ebenfalls an Pin 3. Die Katode von D1 wird an IC25 Pin 9 angeschlossen, die Katode von D2 wird über ein Kabel mit IC 32 Pin 27 verbunden.

Schließlich werden die nötigen Brücken eingebaut, falls nur ein Teil der Peripheriebausteine bestückt ist. Das sind bei IC 36 (SIO) Pin 6 und 7, bei IC 37 (PIO) Pin 22 und 24. Es bleiben noch die Stecker einzubauen, dann ist man fertig.

(wird fortgesetzt)

Geschenkt!

Die im Ausschnitt gezeigte Tabelle, eine der sogenannten mc-Arbeitshilfen, erhalten Sie geschenkt, wenn Sie sie beim Verlag, Abt. Vertrieb, bestellen. Die hier gezeigte Arbeitshilfe enthält die Zeichencodes, die in gebräuchlichen Computern vorkommen. Damit werden Konvertierungen zum Kinderspiel oder Treiber-Patches überhaupt erst möglich.

Tabelle 5: Die Bauteile der Z80-Karte

Anzahl	Typ	Bezeichnung	Bemerkungen	Inverse Relation:	
1	2764	IC 33	oder 27128, siehe Tabelle 1	IC 1	74LS157
8	4164	IC 2...9	siehe Tabelle 1	IC 2...9	4164
1	74LS00	IC 17		IC 10	74LS122
1	74LS05	IC 19		IC 11	XTAL
1	74LS14	IC 15		IC 12	74LS74
1	74LS32	IC 14		IC 13	74LS157
5	74LS74	IC 12, 21...24		IC 14	74LS32
1	74S112	IC 20	kein LS-Typ	IC 15	74LS14
1	74LS122	IC 10		IC 16	–
1	74125	IC 30	kein LS-Typ	IC 17	74LS00
1	74LS132	IC 18		IC 18	74LS132
1	74LS139	IC 26		IC 19	74LS05
2	74ALS257	IC 1, 13	oder 74AS157	IC 20	74S112
2	74LS244	IC 25, 29		IC 21	74LS74
2	74LS245	IC 27, 31		IC 22	74LS74
1	Z80-CPU	IC 32	siehe Tabelle 1	IC 23	74LS74
1	Z80-CTC	IC 35	siehe Tabelle 1	IC 24	74LS74
1	Z80-PIO	IC 37	siehe Tabelle 1	IC 25	74LS244
1	Z80-SIO	IC 36	-0/-1/-2, Tab. 1	IC 26	74LS139
1	XTAL	IC 11	Quarzosz. Tab. 1	IC 27	74LS245
2	1N4148	D1, D2			
0...5	Stiftleisten	ST2...6	zweireihig 2 x 10pol.	IC 28	–
1	DIN41612	ST1	VG-Steckerleiste 96pol.	IC 29	74LS244
1	100 pF	C 24	} MKS (Styroflex) z. B. SIEMENS B31063	IC 30	74125
1	330 pF	C 18		IC 31	74LS245
1	1 nF	C 19		IC 32	Z80-CPU
1	3,3 nF	C 1		IC 33	2764/128
28	100 nF	C 2...13, 17,	Abblock-	IC 34	–
		C 20...23, 25...35	kondensatoren	IC 35	Z80-CTC
4	10 µF	C 36...39	Tantal-Kondens.	IC 36	Z80-SIO
1	(Tab. 1)	R 1	Metallschicht 1%	IC 37	Z80-PIO
1	(Tab. 1)	R 2	Metallschicht 1%		
1	(Tab. 1)	R 11	Metallschicht 1%		
1	330 Ω	R 4			
8	3,3 kΩ	R 3, 5...10, 12	-		
13	IC-Fass.	14pol.	anreihbar		
12	IC-Fass.	16pol.	anreihbar		
4	IC-Fass.	20pol.			
2	IC-Fass.	28pol.	einschl. CTC		
3	IC-Fass.	40polig	einschl. SIO und PIO		
1	Platine	Z80-Karte	DSM		